Reconfigurable Optically Interconnected Systems

Yiwen Shen

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2020

ABSTRACT

Reconfigurable Optically Interconnected Systems

Yiwen Shen

With the immense growth of data consumption in today's data centers and high-performance computing systems driven by the constant influx of new applications, the network infrastructure supporting this demand is under increasing pressure to enable higher bandwidth, latency, and flexibility requirements. Optical interconnects, able to support high bandwidth wavelength division multiplexed signals with extreme energy efficiency, have become the basis for long-haul and metro-scale networks around the world, while photonic components are being rapidly integrated within rack and chip-scale systems. However, optical and photonic interconnects are not a direct replacement for electronic-based components. Rather, the integration of optical interconnects with electronic peripherals allows for unique functionalities that can improve the capacity, compute performance and flexibility of current state-of-the-art computing systems. This requires physical layer methodologies for their integration with electronic components, as well as system level control planes that incorporates the optical layer characteristics. This thesis explores various network architectures and the associated control plane, hardware infrastructure, and other supporting software modules needed to integrate silicon photonics and MEMS based optical switching into conventional datacom network systems ranging from intra-data center and high-performance computing systems to the metro-scale layer networks between data centers. In each of these systems, we demonstrate dynamic bandwidth steering and compute resource allocation capabilities to enable sig-

nificant performance improvements. The key accomplishments of this thesis are as follows.

In Part 1, we present high-performance computing network architectures that integrate silicon photonic switches for optical bandwidth steering, enabling multiple reconfigurable topologies that results in significant system performance improvements. As high-performance systems rely on increased parallelism by scaling up to greater numbers of processor nodes, communication between these nodes grows rapidly and the interconnection network becomes a bottleneck to the overall performance of the system. It has been observed that many scientific applications operating on high-performance computing systems cause highly skewed traffic over the network, congesting only a small percentage of the total available links while other links are underutilized. This mismatch of the traffic and the bandwidth allocation of the physical layer network presents the opportunity to optimize the bandwidth resource utilization of the system by using silicon photonic switches to perform bandwidth steering. This allows the individual processors to perform at their maximum compute potential and thereby improving the overall system performance. We show various testbeds that integrates both microring resonator and Mach-Zehnder based silicon photonic switches within Dragonfly and Fat-Tree topology networks built with conventional equipment, and demonstrate 30-60% reduction in execution time of real high-performance benchmark applications.

Part 2 presents a flexible network architecture and control plane that enables autonomous bandwidth steering and IT resource provisioning capabilities between metro-scale geographically distributed data centers. It uses a software-defined control plane to autonomously provision both network and IT resources to support different quality of

service requirements and optimizes resource utilization under dynamically changing load variations. By actively monitoring both the bandwidth utilization of the network and CPU or memory resources of the end hosts, the control plane autonomously provisions background or dynamic connections with different levels of quality of service using optical MEMS switching, as well as initializing live migrations of virtual machines to consolidate or distribute workload. Together these functionalities provide flexibility and maximize efficiency in processing and transferring data, and enables energy and cost savings by scaling down the system when resources are not needed. An experimental testbed of three data center nodes was built to demonstrate the feasibility of these capabilities.

Part 3 presents Lightbridge, a communications platform specifically designed to provide a more seamless integration between processor nodes and an optically switched network. It addresses some of the crucial issues faced by the works presented in the previous chapters related to optical switching. When optical switches perform switching operations, they change the physical topology of the network, and they lack the capability to buffer packets, resulting in certain optical circuits being unavailable. This prompts the question of whether it is safe to transmit packets by end hosts at any given time. Lightbridge was developed to coordinate switching and routing of optical circuits across the network, by having the processors gain information about the current state of the optical network before transmitting packets, and being able to buffer packets when the optical circuit is not available. This part describes details of Lightbridge which is constituted by a loadable Linux kernel module along with other supporting modifications to the Linux kernel in order to achieve the necessary functionalities.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my advisor, Professor Keren Bergman for her guidance during my PhD career. Her advice and support helped me not only build my knowledge and skills in the research topics presented in this thesis, but also develop a solid foundation in the writing and presentation of my ideas and concepts throughout these years.

I am deeply grateful for my colleagues at the Lightwave Research Lab, especially Payman Samadi who mentored me during my early years of my PhD career and gave me the skillset to become independent in my own research. I would also like to thank other members of the lab including Ke Wen, Sébastien Rumley, Madeleine Glick, Ziyi Zhu, Alexander Gazman, Qixiang Cheng, Maarten Hattink, Alex Meng, Yu-han Hung and many others for their discussion, collaboration and contribution to my research projects. Additionally I would like to thank to Professor John Shalf, Professor George Michelogiannakis, Professor Sheryl Woodward, Professor Dan Kilper, and Ryan Goodfellow for their external collaboration and guidance.

I would like to acknowledge the US Department of Energy and the Advanced Research Projects Agency Energy (ARPA-E) for sponsoring my research, as well as PICA8, Calient and OpSIS whose switching hardware were central components in my projects.

I would also like to thank all the members in the dissertation committee, Professor Keren Bergman, Professor Gil Zussman, Professor Ethan Katz-Bassett, Professor Dan Kilper, and Professor Christine Hendon for evaluating my thesis.

Finally I would like to thank my parents, Hong and Meilin, for their continuous support in all my pursuits.

Introduction

0.1 Communication Growth in High-Performance Computing Systems

The computational capability of high performance computing (HPC) systems is reliant on the ability to parallelize computational resources. With the stalling of Moore's Law where compute power per individual CPU has reached near its maximum, additional increases in performance have relied on using multiple CPU cores and other discrete components to form powerful microprocessors that are integrated with memory, accelerators, storage, and network interconnects [1]. While developments in parallelism through these architectures have produced performance growth, the scalability of these system depends on the underlying interconnection network to continue to provide sufficient bandwidth resources and minimal latency to enable the full computation potential of each processor.

However, over the last few years the growth of interconnect bandwidth capacity has not been able to match the pace of the increase in raw processing power gained through parallelism. The byte/flop ratio, or the ratio of bandwidth between each node to intrinsic processing power of per node of the Top500 [2] top performing high-performance computing (HPC) systems, have decreased by a factor of over 6$\times$ since June 2010 [3] despite

the rising peak computing performance. As of June 2019, the most powerful supercomputer, *Summit*, achieved 200.8 PFlops using 4608 servers, which means each server performs 43 TFlops. However, each node only receives 25 GB/s bandwidth with dual NVLINK bricks resulting in a 0.0006 byte/FLOP ratio [4]. This trend cannot continue indefinitely; such low byte/FLOP ratio means processors that are constantly starved for data to process and therefore are not operating at their full computation potential, resulting in an overall sub-optimal system performance and energy efficiency.

This effect can be seen quantitatively when we observe the performance of HPC systems when operating the High Performance Gradients (HPCG) benchmark instead of the traditionally used High Performance Linpack (HPL) benchmark. *Summit* performed only 2.9 PFlops with HPCG (compared to 148.6 PFlops with HPL). This is because the Linpack program performs compute-rich algorithms to multiply dense matrices, called Type 1 patterns, favoring systems with high computational capabilities [5]. However, most current applications require lower computation-to-data-access ratios, referred to as Type 2 patterns, which require higher bandwidth communications with the memory subsystem [6]. The HPL program focuses only on Type 1 patterns, while the HPCG benchmark has emerged as a new metric for HPC systems that is designed to evaluate systems with both Type 1 and Type 2 patterns. As such, performance of the top ranked supercomputers operating HPCG show much more modest results compared to their HPL counterparts. If we look at the performance in terms of energy efficiency, *Summit* showed 14.7 GFlops/W with HPL but only 0.3 GFlops/W with HPCG. Overall, this large decrease in *Summit*'s performance between the two benchmarks highlights the crucial role that the interconnection network plays in supplying a system's computational resources with sufficient

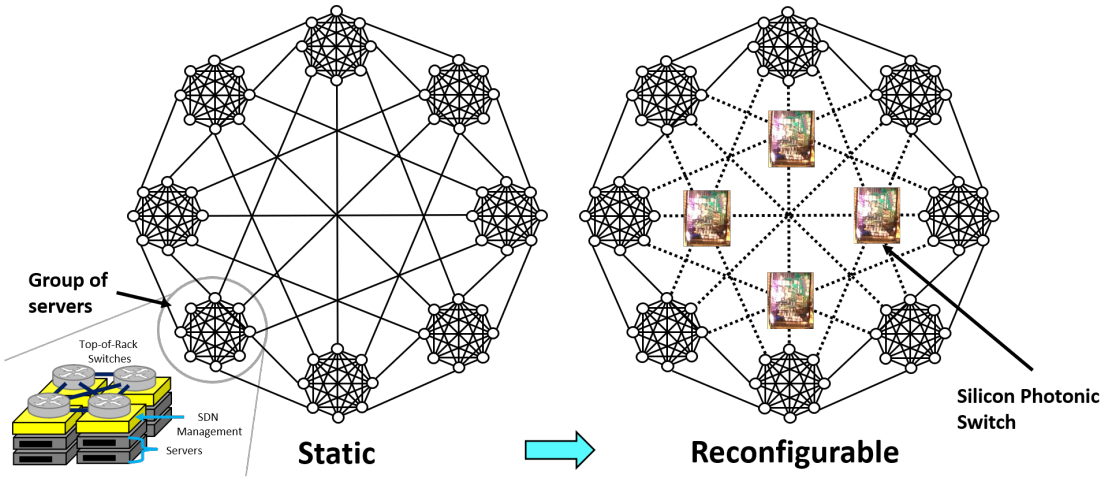


Figure 0.1. Transforming the highly connected static HPC network into a reconfigurable network by integrating SiP switches in between global inter-rack links

traffic to memory and other computing nodes in order to bring forth the system's full performance potential.

In order to address the growing challenge of providing high bandwidth capacity network interconnects, today's HPC systems follow a best-for-all approach that is characterized by over-provisioned, static networks, with components with high communication placed close together [7]. However, such an approach is expensive and inefficient, and lacks the scalability to meet the rising demand in computational power [3]. This is because a majority of HPC applications produce skewed traffic patterns over the network infrastructure, where the traffic demand is concentrated within only a small percentage of the total available links, and this traffic characteristic varies slowly over the entire runtime of the application [8]–[10]. Each application feature a characteristic traffic matrix, and operating the application with such a traffic pattern on a general-purpose, static network results in both congested links that limit the rate in which data feeds the processors, as well as under-utilized links wastes energy usage [11].

This mismatch between the application traffic distribution and the bandwidth allocation of the physical infrastructure presents the opportunity to build adaptive networks that can reconfigure the physical layer network topology to optimize bandwidth resources between highly intensive communicating nodes. This can be achieved by integrating optical circuit switching through the means of microelectromechanical systems (MEMS) switches or silicon photonic switches within the traditional network infrastructure of electronic packet switches (EPSs). These optical switches can be controlled to change the network endpoints that of optical links, which allows the network to dynamically steer the bandwidth of available links to match an application's unique traffic demands, in a process called bandwidth steering. This concept is illustrated in Figure 0.1. The management of these switches can be performed with a modified software-defined networking (SDN) control plane combined with external hardware such as Field-Programmable Gate Arrays (FPGAs) for sending electronic bias signals to the photonic switches. On the network layer, the SDN control application has knowledge of the global network traffic characteristics, and can manage the routing tables in all Layer 2 or Layer 3 (L2/L3) electronic packet switches (EPSs) as well as control the configuration of the optical switches.

0.2 The Need for Photonic Interconnects

Energy Challenges and Opportunities

Optical interconnects have become the leading technology to support future-proof data rates with extremely high energy efficiency. A single optical fiber or silicon waveg-

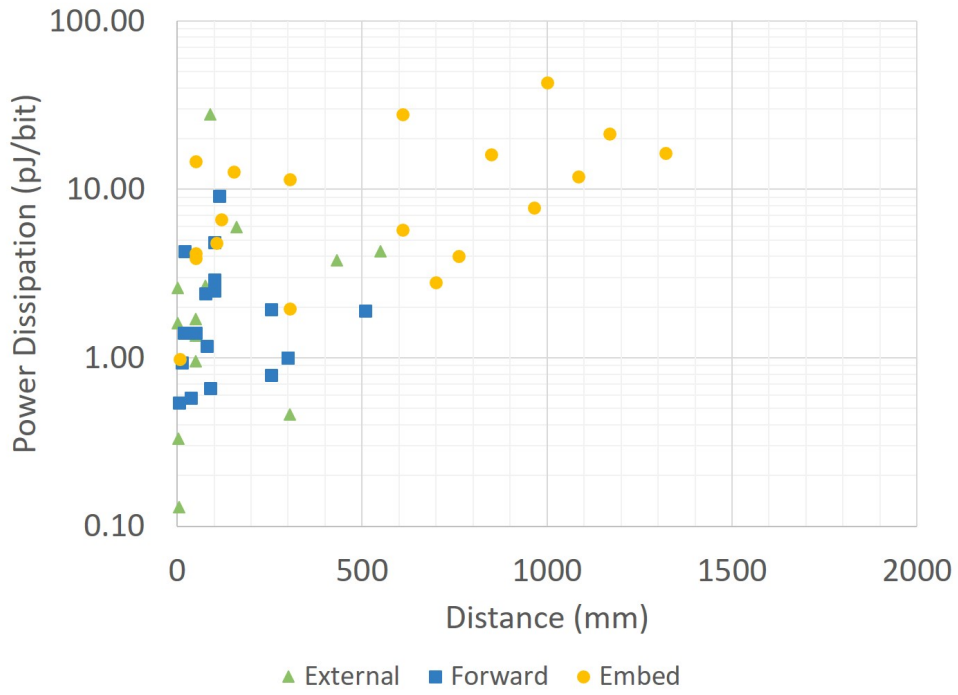


Figure 0.2. Power dissipation plotted over distance for electronic interconnects [12]

uide can terabit-scale bandwidths through the means of wavelength division multiplexing (WDM) over large distances. While there has been considerable work for electronic interconnects to provide sub-pJ/bit energy efficiencies over gigabit data rates [12], such performance is only possible on a chip-scale range (under 300 mm). Figure 0.2 shows the energy efficiency over distance for state-of-the-art electronic interconnects, categorized by clocking architecture (external clock, forward, or embedded). It can be observed that as distances increase beyond 300 mm, energy consumption increases to tens of pJ/bit or greater, meaning that conventional electronic links operating at high frequency have difficulty maintaining sub pJ/bit efficiencies due to physical effects such as the skin effect and the dielectric loss, both of which scales with frequency. Electrical interconnects are also limited by wiring density, which is worsened by smaller cross-sectional sizes due to higher resistive and capacitive effects [13].

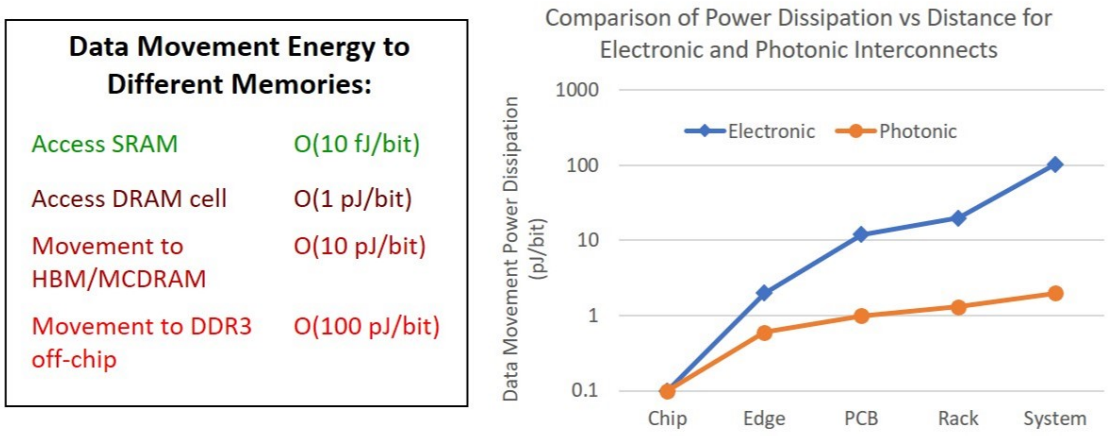


Figure 0.3. Power dissipation of data movement to different memory types (left) and different distances (right) [14]–[16]

These physical limitations have given rise to the ”bandwidth taper” phenomenon, which refers to the orders of magnitude decrease in bandwidth provided by electronic interconnects as data travels over distances ranging beyond on-chip scales, to off-chip, inter-node and inter-rack [14], [15]. This translates to a proportional increase in energy consumption, as displayed in Figure 0.3 which lists the energy requirements for data movement to different types of memory types and to varying distances [17].

The general system requirements for optical interconnects are approximately 1 pJ/bit for rack-to-rack distances, and less than 0.1 pJ/bit for chip-to-chip scales. The energy consumption of an optical link depends primarily on the laser source, data modulation and its associated drivers. After the data has been transformed into its optical format and transmitted onto the waveguide or fiber, the losses, while still distance dependent, are much lower than electrical interconnects, and they are not data rate dependent. The break-even distance where optical interconnects become more power efficient than their electrical counterparts is approximately 1 mm at a data rate of 10 Gbps [15].

Recent developments in optical interconnects have enabled silicon photonics (SiP) to

transform existing network design paradigms. With recent developments that enable photonic components to be tightly integrated with electronic processing and memory peripherals [18], [19], SiP interconnects can potentially enable truly scalable extreme-scale computating platforms and opportunities for ultra-low energy transmission of close-proximity electronic driver circuitry integrated on-chip.

Optical Switches

In this thesis the primary form of optical interconnect used are silicon photonic and MEMS switches, which are used for redirecting optical signals carrying high bandwidth traffic that are traveling on commercial optic fibers that are coupled into the switch. All optical transmission or modulation is performed with commercial transceivers. In Part 1 of this work, both Mach-Zehnder Interferometer (MZI) and microring-resonator (MRR) based SiP switches are used for intra-rack switching. Part 2 will feature commercial MEMS switches in the testbed for inter-data center switching. Part 3 applies to all forms of optical switches.

Silicon photonic (SiP) switches have the advantages of low power consumption, small area footprint, low fabrication costs at large scales, and the potential for nanosecond range dynamic connectivity [20]. Their physical operation is based on the strong thermo-optic (T-O) coefficient (1.8×10^{-4} /K) of silicon which can be leveraged to tune the phase of light passing through the switch in timescales of tens of microseconds, as well as using the plasma dispersion effect through carrier injection or depletion for nanosecond scale switching times.

From the point of view of the network operator integrating SiP switches for server or rack scale interconnects, the means to which an SiP switch (whether MZI or MRR type) performs its switching operation can be treated as a black box for the most part. Both types of switches also use the same control mechanism - electronic bias signals input to drive the thermo-optic heaters or electro-optic drivers for carrier injection/depletion. However, the fundamental difference between MZI and MRR SiP switches that network operators needs to be aware of is their wavelength bandwidth characteristics. MZI switches are broadband switches, and their bandwidth spans nearly the entirety of the C-band or O-band (depending on fabrication parameters), while MRR switches are narrowband devices that will only allow for a small range of wavelengths to pass (typically only a single channel), while other wavelength channels are supressed. Therefore, when using signal sources that feature multiple channels, such as the commercial 100G QSFP28 CWDM4 transceivers, in which a single fiber contains four 25 Gbps WDM signals at typical wavelengths of 1271 nm, 1291 nm, 1311 nm, and 1331 nm, an MRR switch would not be a viable choice as a single MRR would not cover all these channels, but an MZI switch may be possible. However, if the signal source was from a commercial SFP+ transceiver which is a single 10G signal at a particular wavelength, then MRR switches are a viable choice.

Compared to free-space and silica-based switches such as 3D-MEMS or liquid crystal optical switches, both MZI and MRR SiP switches have the disadvantages of fewer ports and higher losses due to challenges in coupling, crosstalk, and polarization [21]. The current record for thermo-optic silicon switches is a 64×64 implementation of MZI elements in the Beneš topology [22], while 3D-MEMS switch can easily have hundreds

of ports, such as this 512×512 port switch from [23]. These switches are also lower loss, polarization insensitive and are broadband. However, SiP switches, which have yet to be commercialized and still under a research phase, are promised to provide high density, low power, much faster switching, and much lower costs in bulk, making them a much more sensible option for future on-chip interconnects.

0.3 Flexible Metro-Scale Distributed Data Center Networks

The capability for networks to be reconfigurable and enable remote management is fundamental to next generation inter-data center networks as well. Metro data centers are undergoing drastic transformations in order to meet the predicted explosive demand of traffic over the next few years, driven by newly emerging technologies including a seven-fold increase in mobile data traffic and 12-fold increase in virtual reality and augmented reality traffic from 2017 to 2022, as well as other applications such as Internet video to TV, video surveillance traffic, gaming, coinciding with the advent of 5G and the Internet of Things [24]. Altogether, it is predicted that global IP traffic will triple to 4.8 zettabytes by 2022 from 1.5 zettabytes during 2017, and a third of this will be carried by the metro-capacity of total service provider network capacity [25].

To meet this demand, enterprises and cloud-providers have drastically increased the deployment of data centers. Due to the construction and maintenance costs of managing data centers as well as the increased adoption of content distribution networks (CDNs) us-

ing edge data centers, enterprises are moving towards deploying small- to mid-sized data centers to bring data and services closer to the user to reduce latency [26], [27]. These data centers are separated by metro-scale distances and are connected with a fiber network. Such cloud-based and 5G systems will place strict demands on the network infrastructure to support specific levels of quality of service (QoS) in terms of bandwidth and latency requirements. For example, migration of virtualized Evolved Packet Cores (vEPCs), virtual machines and many virtual reality gaming applications require strict end-to-end latencies [28]. Other applications such as high definition television require large bandwidths. Like HPC systems, current inter-data center networks also use static, overprovisioned networks. In this case, adding or removing a new optical connection in the physical layer can take days or weeks to occur. With the use of software-defined networking (SDN) having global awareness of traffic throughput and potential hotspots, and using optical switches to allow physical topology reconfiguration, we enable remote management of network components in the data plane from the higher layers. This allows for dynamic networks that can support connections with different bandwidth and latency requirements on the fly. Additionally, it provides the capability for the system to be autonomous and less error-prone than human-operated systems.

Part I

Bandwidth Steering for

High-Performance Computing Systems

Chapter 1

Network Architecture for Photonic Switch Integration

1.1 Introduction

The emergence of data-intensive high-performance computing (HPC) applications where large quantities of data needs to be processed and communicated have caused the network infrastructure of these systems to become a bottleneck to overall system performance. This is due to the fact that as more and more nodes are used to parallelize workloads, common operations such as gathering and reducing as well as general coordination between nodes cause communication among nodes to increase. Current systems use a best-for-all approach characterized by over-provisioning static links with highly-connected topologies [7]. However, many HPC applications are characterized by skewed communication patterns that concentrate traffic within only a small percentage of total available links, resulting in both under-utilized links that wastes power as well as over-subscribed links that limit system performance [9]. This is explicitly shown in the traffic matrices of various HPC applications depicted in Figure 1.1, which were obtained in simulations on SST/macro [9]. The x and y axis represent the source and destination node groups, and the various colors indicate the intensity of traffic between these nodes. As can be seen, each application has a unique pattern. In each one there are significant parts of the

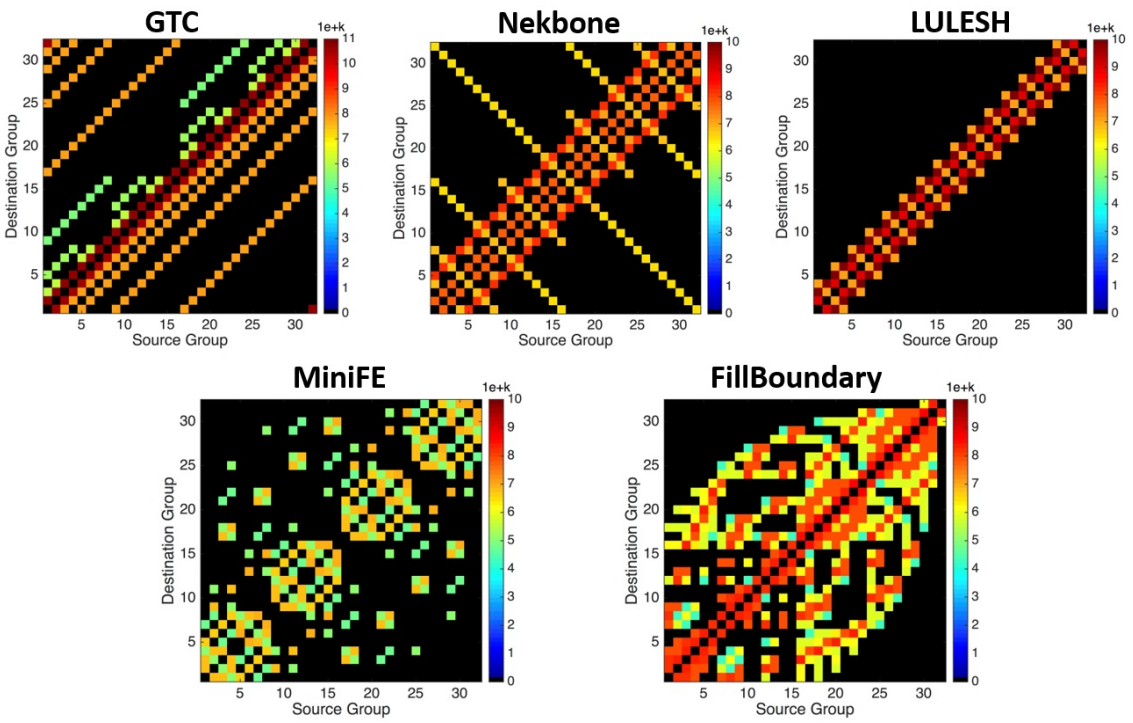


Figure 1.1. Traffic matrices of various HPC applications, showing the intensity of traffic between different groups of nodes [9]

matrix that is black, indicating that no traffic occurs between these nodes at all, and any links connecting these nodes would be wasting energy. At the same time, the diagrams show that these examples of HPC applications feature heavy direct neighbor-to-neighbor traffic, which means the links between neighboring groups are likely to be congested.

For current fixed topologies, this mismatch in traffic distribution and system bandwidth allocation is mitigated through global or adaptive routing. Numerous routing strategies have been proposed [29]–[35], but in general these result in longer-distanced paths and cross-group interference [36], [37]. Instead, our approach is the development of flexible, reactive networks that can utilize the available network resources optimally depending on the traffic characteristics. This is enabled by the advent of software-defined networking (SDN) which disseminates control plane intelligence to the physical network

entities from the higher layers. At the physical layer, novel interconnect technologies such as silicon photonics (SiP) enable the movement of large amounts of traffic while providing low power consumption and low fabrication costs at large scales with CMOS compatibility. While the integration of silicon photonic switching technologies into basic network systems has been shown with experimental prototypes [9], [38]–[41] and theoretical works that examine the scalability of SiP switches [42], [43], there has been little study on a control plane to integrate and synergistically utilize SiP switching within a large DC or HPC packet-switched network environment.

In the past there have been many efforts in software-defined elastic optical networks (SD-EONs) which feature a similar control plane architecture to this work, but integrates bandwidth variable wavelength selective switches (BV-WSSs) instead of SiP switches in the data plane to allocate variable spectrums for dynamic optical path provisioning [44]–[47]. While the control planes for using BV-WSSs to allocate wavelength/spectrum are well-developed, most research efforts into control planes for SiP switches still remain at the physical layer, and the arbitration strategies required to utilize SiP switches in a large scale experimental Datacom testbed has been a relatively unexplored area. We attempt to address this by developing a scalable SDN control plane to utilize SiP switches to perform optical circuit switch based bandwidth steering on a system level testbed operating real traffic from open-source HPC benchmark applications.

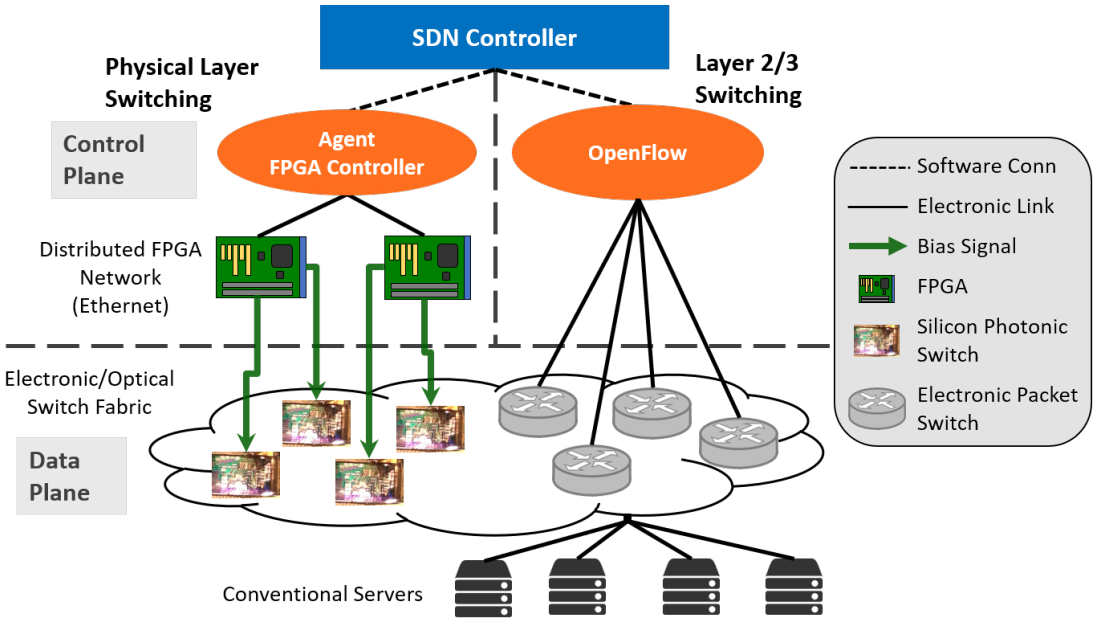


Figure 1.2. Network architecture divided into the control plane and data plane. The SDN Controller manages all the active switching elements in the system, including the SiP switches at the physical layer and the EPSs at Layer 2/3 switching

1.2 Control Plane Overview

In this section the control plane and data plane that integrates SiP switches for optical circuit switching within a conventional datacom environment is presented. Interfaces between various software components with data plane hardware devices are described [48]–[50].

Control Plane Overview

The overview of the network architecture describing the integration between the software control plane and the electronic packet switches (EPSs) and silicon photonic (SiP) switches and servers is shown in Figure 1.2. Beginning at the top is the central management component - the software-defined networking (SDN) controller, which manages the behavior of all the active switching elements in the network. From there, the network architecture

is divided horizontally into the control plane and data plane layers, and vertically into the physical layer switching and layer 2/3 switching parts. The data plane layer consists of the electronic and optical switch fabric which are composed of the SiP elements and electronic packet switches. Servers are connected to the EPSs first, and the SiP switches are placed in-between EPS connections to serve as dynamic inter-rack connections.

During a switching operation, both the EPS and SiP switch must work in tandem to create an end-to-end path for packets to travel between nodes. While the SiP switch creates the physical route, the EPS must update its routing table to reflect the new change in the topology. Otherwise, even if the packet is able to travel through the optical switch and reach the endpoint, the packet switch will drop this packet if it has no flows instructions to handle this packet. To synchronize the operations of the SiP switch and EPS, the SDN controller is designated to be the central point of management that sends a command to both switches in parallel to minimize the delay between the switching modality of the electronic packet switches and SiP switches.

1.3 Layer 2/3 Electronic Packet Switching

The routing tables in the EPSs dictates the path taken by TCP/IP packets in between the servers. Their behavior is managed by the SDN controller through the OpenFlow protocol, which is a standard southbound application programming interface (API) protocol used commonly by SDN applications [51]. The SDN application plays the role of adding or deleting layer 2 or layer 3 flow rules to the flow tables of the EPSs. Layer 3 flow rules consist of the in-port and out-port values, as well as the IP source or destination address.

These three fields determine the route that is taken by the packets from the source node to the destination node. As mentioned previously, since the SiP switches modify the physical topology of the network and connects different endpoints together that change over time, the SDN controller must be able to add/delete flows to coordinate the correct in-port and out-port values as well as the IP destination address with the topological change caused by the SiP switch. If the routing table has not been updated in time or correctly, the packets will be dropped by the EPS connected to the destination compute node, as its fields do not correspond to any known flows in its routing tables.

Each source-destination node pair that is the minimum distance (1 hop) from each other requires 4 individual flows in the flow table - the IP flow, ARP flow, and the same flows but with the in-port and out-port flipped and a different IP destination for the reverse direction as to make the connection bidirectional. These flow rules on the EPS allow the switch to forward IP and ARP traffic, both of which are required for two servers to communicate with each other. The first packet sent by a source server is the broadcast type ARP request to learn the MAC address of the server corresponding to the destination IP address, and only then can the two servers exchange IP data packets. A hop refers to the connection between EPSs, which may or may not span across the SiP switch. Each additional hop to another EPS in the path requires 4 additional flows.

The number of bytes required to send a single flow modification message to the EPS averages around 150 bytes, which we found using Wireshark [52] which tracks individual packets over a network. A flow table modification is performed by sending the OpenFlow FlowMod message type from the SDN controller to the EPS, whose frame structure is outlined in [53]. This includes various key fields, namely:

- *Header* (4 bytes) - identifying as the type of packet
- *flow priority* (2 bytes) - determines the order of the filter, where higher priority priority flows are used instead of lower priority flows if the two flows have overlapping matches
- *out_port* (4 bytes) - representing the outgoing port on the EPS
- *match* (4 bytes) which can consist of various fields that match incoming packets to the flow (such as *in_port* and *source* and *destination* IP addresses
- *instruction* (4 bytes) for how to route an incoming packet (such as providing an out_port value

1.4 Physical Layer Silicon Photonic Switching

Control of the SiP switch(es) were performed through interfacing the SDN controller application southbound with a distributed FPGA network, which is capable of directly applying pre-defined voltage bias signals to the SiP switches. The network used for communicate messages by the SDN application is the general campus Internet connection, which is a 1G out-of-band Ethernet network. By using an Ethernet network for the control mechanism as opposed to a serial network, we future-proof the system for scalability. If additional SiP elements due to additional switches or larger radix switches are used that exceeds the control capabilities of the current available numbers of FPGAs, additional FPGAs can simply be connected to an off-the-shelf Ethernet router, making the system easily adaptable to scale. Details of the control mechanism is shown in Figure 1.3.

Each FPGA unit is equipped with Digital-to-Analog Converter (DAC) chips, which

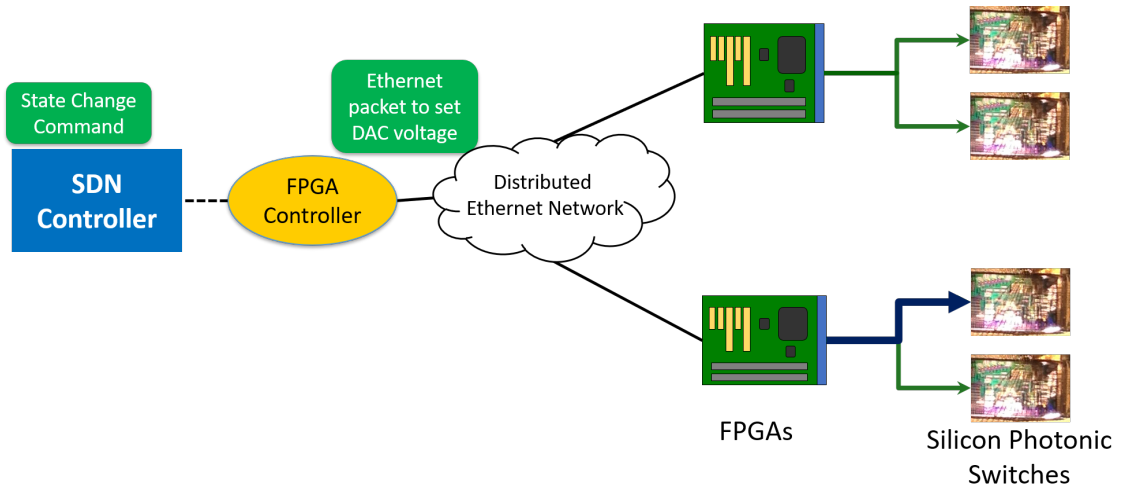


Figure 1.3. SiP switch state change command being sent from the SDN controller, and converted to an Ethernet packet to arrive at the specific FPGA in the distributed FPGA network to control a given SiP switch

allows per-defined voltages/currents to be applied to the SiP switches. For the SDN application to control the FPGA network, an in-house C++ application was developed, called the FPGA Controller. The FPGA Controller acts as an intermediary to allow the SDN controller to communicate with the FPGA microcontroller. To do this, the SDN controller application first connects to the FPGA Controller with a TCP socket connection. During a physical topology change, the SDN application simultaneously modifies both the flow rules on the EPS and sends a flow mod message to the FPGA Controller to set a given SiP switch to a specific configuration by sending which input and output port numbers of the SiP are to be connected. The format of this message is described in the following section. When the FPGA Controller application obtains the message from the SDN controller, it prepares a flow update Ethernet packet that contains the input and output port numbers to the FPGA unit that controls the relevant SiP element. Once the packet is received by the FPGA microcontroller, the device port numbers that are to be connected

are read and mapped to a set of pre-defined voltages hardcoded into individual registers of the FPGA. These voltage values are then applied to the DACs of the FPGA in order to bias the SiP switch to change its configuration. A 5× amplification is required to deliver enough power the cause the SiP element to perform the state change. This amplification is performed by a DAC gain stage implemented on a printed circuit board (PCB). The amplified bias signals maintain their voltage levels until a new state change command from the SDN controller has been received.

While the in-house FPGA Controller software serves the purposes to communicate with the FPGA network to control SiP switches in this work, in future works an OpenFlow client for the FPGAs can be developed so that the control plane unifies its southbound APIs to the EPSs and the SiP switches to improve compatibility and scalability. This will also enable further reductions in latency as the SDN controller will be able to directly communicate with the FPGA without an intermediary module. Additional performance improvements can be done by integrating ARM processors to run the OpenFlow client and use hardware to decode extracted flow updates and offloading the tasks needed to be performed by the FPGA CPU. This can raise the performance and reduce latency in processing flow modification commands to what is achievable by application-specific integrated circuits (ASICs) without significantly raising costs or limiting reconfigurability options.

Silicon Photonic Switch Control Messaging Protocol

A packet protocol under the Ethernet II frame format was developed to provide communication between the SDN controller and the distributed FPGA network nodes. This

common frame format allows for ease of use and scalability as additional FPGAs can be integrated into the system by connecting it to an Ethernet switch that connects all the FPGAs to the SDN controller.

The format of the custom Ethernet frames is presented below in Table 1.1 and 1.2. Two types of custom protocols are used: the FPGA Ethernet Protocol and the Register Ethernet Protocol. The FPGA Ethernet Protocol is for communication between nodes (server to FPGA, or between FPGAs), and contains rudimentary features to handle packet loss using the Message ID field to allow for retransmission. The Register Ethernet Protocol is used during reading to or writing from the FPGA's memory-mapped registers. The data from the Register Ethernet Protocol is placed in the payload section of the FPGA Ethernet packet. Inside the Register Ethernet Protocol, the payload section contains two integers, each contained within 8 bits, that represent the input and output port values of the SiP switch that are to be connected.

The number of bytes used to communicate between the FPGA Controller and FPGAs using the FPGA Ethernet Protocol has a minimum size of 60 bytes (for a single pair of input and output ports), not counting the frame check sequence (FCS) that is 4 bytes long. 8 bytes are needed for every additional pair of input and output ports in the same packet.

FPGA Operation Workflow

The procedures performed by the FPGA after receiving a flow update packet are depicted in Figure 1.4. During the initialization phase, the voltage values associated with each possible configuration of the SiP switch is hardcoded into the FPGA's registers. Once the

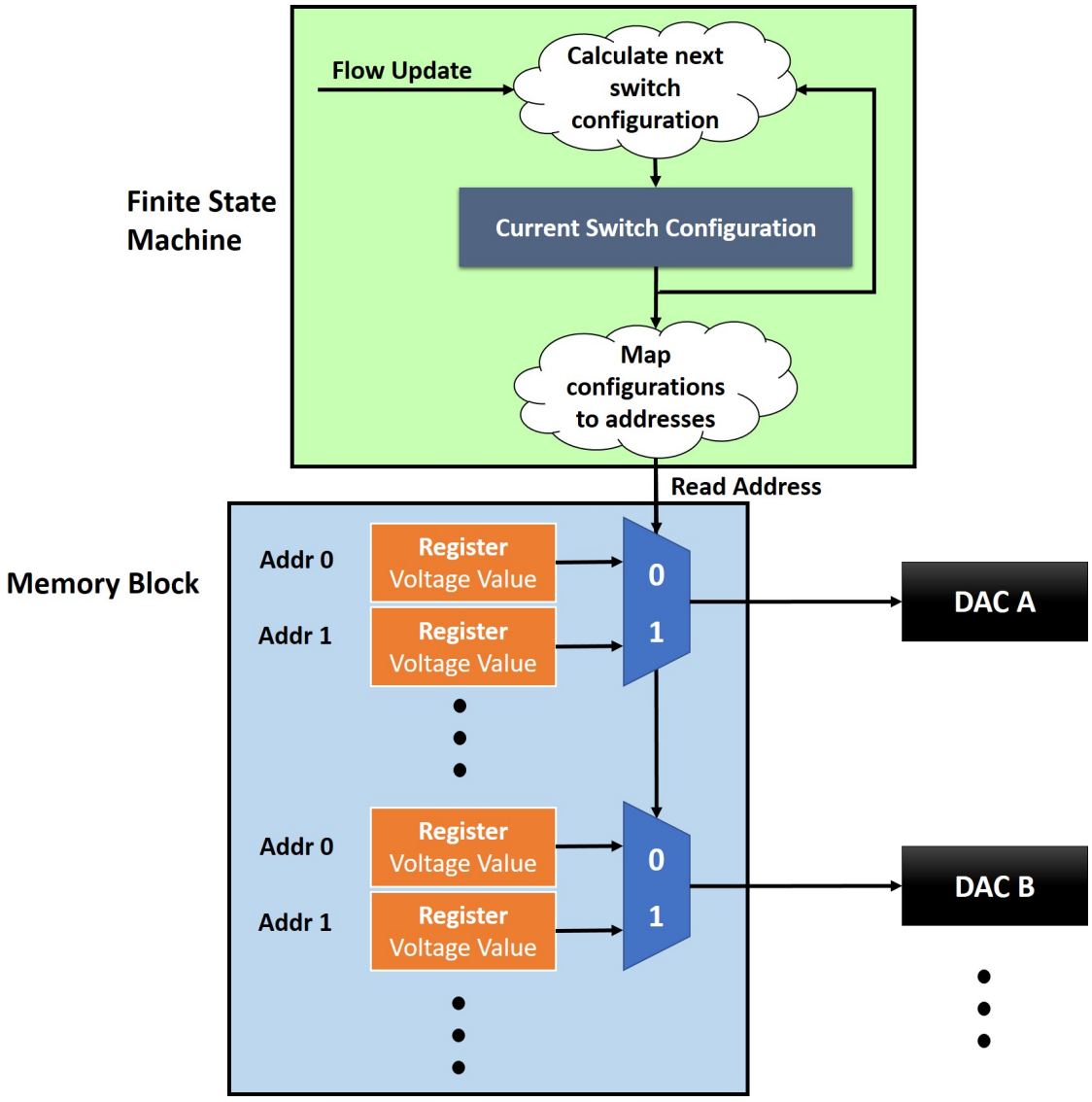


Figure 1.4. FPGA operation workflow for handling an incoming flow update packet from the FPGA Controller

Table 1.1. FPGA Ethernet Protocol

Byte Position	Bit Position	Field Name	Description
0 to 13		Ethernet header	Source MAC Address and the Destination MAC Address, plus the EtherType.
14	7	REQ/ACK Bit	Indicates if a message is a request or response (=0) or an acknowledgement to such a request or response (=1).
14	6 to 0	Message Type	Indicates the type of protocol that send the packet. The Register Ethernet protocol uses the value 0.
15 to 17		Message ID	The sequence number of the message. Both communicating nodes keep a counter for both the other's message ID and their own. Each new transaction raises the counter by 2.
18 to 1514 (max)		Payload	

Table 1.2. Register Ethernet Protocol

Byte Position	Bit Position	Field Name	Description
0 to 1		Address	Address being written to or read from. The register interface is 32 bit addressed.
2 to 3	0 to 13	Count	Number of 32 bits being read or written or sent as a response in this command.
3	6 to 0	Response Bit	1 if the command is a response type, else 0.
3	7	Write Bit	1 if command is a write type, else 0.
4 to 4+(4*Count-1)		Payload	In case of a WRITE or RESPONSE command the 4 byte command is followed by the data.

FPGA receives flow update packets from the FPGA Controller, it parses the port numbers that are to be connected and uses it to determine the new switch configuration, which are saved using flip-flops so that it functions as a Moore finite state machine (FSM). From there, the switch configuration is mapped to the addresses of the registers that contain the voltage values associated with the new switch configuration. Therefore, the FSM is used to select the corresponding registers whose voltage values are then read in parallel and applied to the DACs. DAC voltage values can be read in parallel because separate memory registers are used to hold the voltage values (see Figure 1.4). In the figure shown, two registers containing the hardcoded volt age values are assigned to each, but in general multiple registers containing multiple voltage values can be assigned to each DAC for

larger SiP devices with a greater number of possible configurations. In fact, a single FPGA can be used to house hundreds of configurations given the size of its memory blocks used to hold voltage values in the registers. For example, a typical FPGA holds hundreds to thousands of memory blocks, each of which has tens of kilobits per block, which can be configured to have various widths and depths. If we configure an Altera M20K block which has 20×1024 bits into blocks that are 36 wide and 512 deep, and using 12-bit DACs, samples for 3 DACs and 512 unique configurations can be stored per memory block. With plentiful amount of memory available to store voltage values, we can be assured that voltage values can be stored in separate registers in the memory block which can be read in parallel, so that as the network scales up and more SiP devices are added, the latency for switching will not increase as more SiP elements are required to change configurations.

Control Hardware Scalability

The amount of control hardware required for scaling with larger SiP devices is related to the number of switching elements of the device, which is dependent on the driving scheme, the switch architecture and the port count of the device. Generally, there are two driving schemes for electro-optic Mach-Zehnder type switches - single-armed and push-pull control [54]. Single-armed bias is where only one electro-optic phase modulator of the MZI element is used for switching, and this often requires an additional thermal-optic phase shifter for calibration so that the MZI element is in the cross/bar configuration. In the single-arm drive case, the bias voltage will be V_π to switch to the opposite state. This method requires higher drive voltage and induces higher electro-absorption loss, but it is

favorable as it only requires one DAC to control an MZI element. On the other hand, the push-pull scheme requires to control both arms and thus requires two DACs to control one switch element. The driving voltage required is below V_π.

The architecture of the device determines the number of switching elements. Commonly applied architectures for optical switches include crossbar, Beneš, dilated Beneš, Banyan, N-stage planar, etc. The Beneš architecture requires the least number of switching elements to achieve non-blocking connections for an $N \times N$ switch [55]. The $N \times N$ Beneš switch has a total number of switching elements as $\frac{N}{2}(2\log_2 N - 1)$. Together with the single-arm driving scheme, we can therefore determine the number of DACs to control an $N \times N$ switch scales $\frac{N}{2}(2\log_2 N - 1)$.

Chapter 2

Reconfiguration of the Dragonfly and Fat-Tree Network Topologies

In this chapter two commonly used network topologies in HPC are introduced - the Dragonfly topology and the Fat-Tree topology. Their weaknesses are identified and the way to improve their connectivity and bi-directional bandwidth through the integration of SiP switches within these topologies are discussed.

2.1 Dragonfly Topology

A commonly used network topology for HPC systems is the Dragonfly topology. The Dragonfly topology [56] is a hierarchical topology that organizes the network into groups of top-of-rack (ToR) packet switches, with each ToR switch connecting to server nodes. Within the group, ToR switches are fully connected with each other. Each group is connected to every other group with at least one link. This topology provides high-connectivity with all-to-all global links at the inter-group level to minimize hop count.

The advantages of high-connectivity, however, are diluted by low per-link bandwidth. In particular, the bandwidth of inter-group (global) links, carrying all the traffic between two large sets of routers groups, becomes the most scarce resource and can create a bot-

tleneck for the entire network. The highly skewed traffic characteristic in nearly every HPC application will create a scenario where some of the inter-group links are highly congested, while others are severely underutilized, and remain so for a large portion of the application's runtime, or even for its entirety. Therefore, the static Dragonfly's highly connected yet diluted per-link bandwidths will become a bottleneck for next generation extreme-scale computing platforms [57]–[60].

In response to this observation, we use optical circuit switching by distributing low-radix SiP switches to enable significant performance improvements, transforming the static Dragonfly network to a flexible Dragonfly, or Flexfly [9]. The Flexfly topology does not introduce new bandwidth - it takes under-utilized links and reassigns them to intensively communicating group pairs using bandwidth steering with the SiP switches. This is illustrated in Figure 2.1, which shows the standard Dragonfly topology with all-to-all connectivity on the left reconfigured to focusing on more neighbor-to-neighbor connectivity on the right. In general, the bandwidth steering concept rests upon the assumption that there are both congested links and under-utilized links at the same time during an application run, which occurs due to a skewed traffic pattern. If an application's traffic is evenly distributed across the entire network, then bandwidth steering would not be able to provide any performance benefit.

The insertion of SiP switches allows us to take advantage of its near data rate transparent property to perform bandwidth steering on the large quantities of aggregate WDM traffic of entire Dragonfly router groups per each input port. As the optical switch performs its switching functions purely on the physical layer, it is transparent to higher network layers and applications, which reduces the latency, energy consumption, and

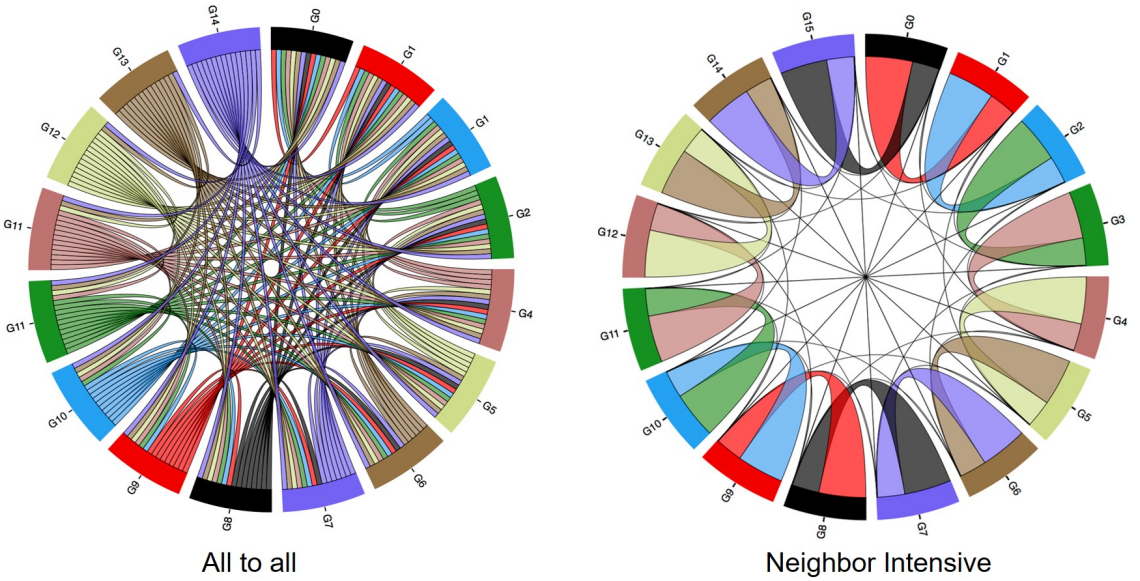


Figure 2.1. Standard Dragonfly topology with all-to-all inter-group links (left) and reconfigured topology after bandwidth steering focusing on neighbor-intensive traffic (right)

Table 2.1. Number of switches and connectors per blade for different G (Dragonfly groups) and r (groups per cabinet row)

G	r	# of supergroups	# of switches	# of connectors
8	4	2	7	56
16	4	4	15	120
16	8	2	15	240
32	4	8	31	248
32	8	4	31	496

routing complexity compared to performing the same task in with another electronic packet switch. Additionally, unlike previous optical switching solutions that rely on large port counts [61], [62], Flexfly is designed to support the use of low-radix silicon photonic switches, realizable through low-cost CMOS fabrication technology.

Figure 2.2 shows how the physical connections of the network topology (first five plots) approaches the traffic matrix of the GTC benchmark application (last bottom right plot), becoming increasingly optimized for this application's traffic characteristics. With

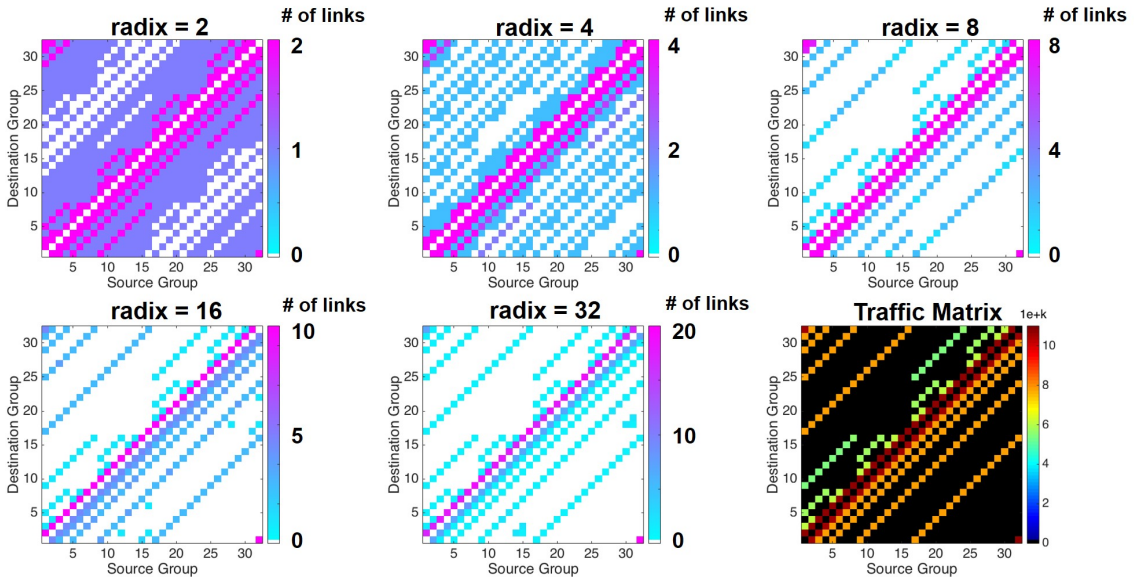


Figure 2.2. The first five traffic matrices shows the physical network topology adapting to the traffic matrix of the GTC application [63] (shown at bottom right) with increasing silicon photonic switch radices.

increasing switch radix, more links can be connected to the photonic switch and resulting in the network becoming more flexible as a whole. As can be seen in the bottom right of Figure 2.2, the traffic matrix for the network at a switch radix of 32 is identical to the traffic matrix of the GTC application [63]. Although it seems that there is little difference in the traffic matrix diagrams of radix = 8 to radix = 32, the colors for each graph are normalized despite different number of links. With a switch of radix = 8 there are a maximum of 8 links available to be steered from under-utilized connections (any white squares) and be given to the most intensive traffic shown at the diagonal. With a switch of radix = 32 however, there are 20 links available to be allocated to the most intensive traffic, which means that not only more bandwidth can be steered, it also allows the network controller to have finer granularity in its resource allocation.

The scalability of the Flexfly architecture is described as follows: We divide a total of

G Dragonfly groups into r groups per cabinet row, resulting in $\frac{G}{r}$ supergroups. A Flexfly switch blade is associated with each supergroup, which contains all the SiP switches associated with the inter-group links of that supergroup. This switch blade will have $G-1$ switches. With r groups per supergroup, each supergroup will have $r(G-1)$ links that fully connect to $G-1$ switches, each with r ports, and $2r(G-1)$ fiber connectors. Table 2.1 shows the number of switches and connectors per blade for different G and r values. The compatibility of SiP with CMOS foundries allows for a large number of SiP switches to be placed on a single chip. Looking at the values shown in Table 2.1, the cost and space needed for incorporating $\frac{G}{r}$ switch blades in a Dragonfly topology HPC system are deemed to be scalable.

2.2 Fat-Tree Topology

The Fat-Tree topology is a type of multi-tiered interconnect topology that consists of routers placed at different tiers and connected in a tree structure, with the processors are connected at the very bottom layer. Fat-Trees and their variants are extensively used in datacenters [64]–[66] and HPC [67]–[70] due to their favorable wiring properties and their ability to provide full throughput for any traffic pattern, assuming perfect load balancing and full bisection bandwidth (i.e., the higher layers of the Fat-Tree have the same total bandwidth as the lower layers) [71]. A recent example in HPC is the HyperX topology that is similar to a Fat-Tree [67].

In a regular Fat-Tree, the number of links going down to its siblings is equal to the number of links going up to its parent. However, to reduce costs, "bandwidth tapering" is

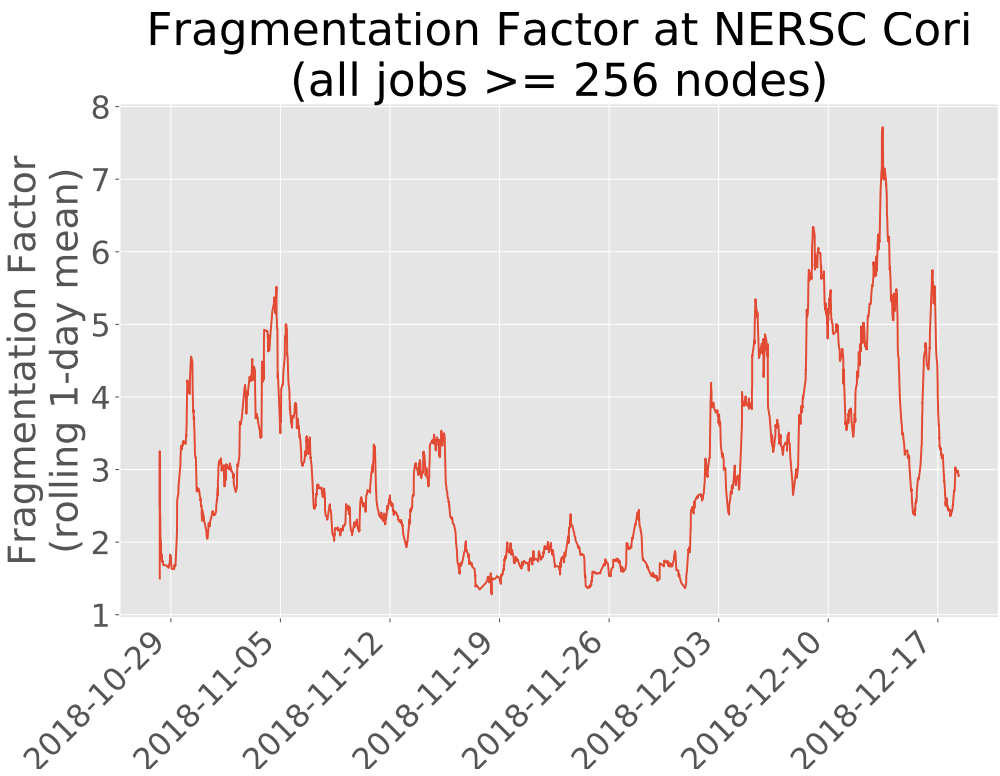


Figure 2.3. Measured data from NERSC's Cori shows that fragmentation is both persistent and dynamic. In this graph, fragmentation is defined as the observed number of Aries (Dragonfly) groups that an application spans, divided by the smallest possible number of groups that the application would fit in

often employed, where the bandwidth of the top-layer links are removed. This is because that bandwidth covers larger physical distances and therefore tends to be more costly [72]. This leads to oversubscription and reducing system efficiency [73]. Examples of large-scale deployed systems using bandwidth tapering include Google's Jupiter topology with 20k nodes that uses a 2:1 oversubscription at the top-of-rack (ToR) uplink layer [66], as well as Microsoft's data center networks which have a 5:1 oversubscription at the ToR layer [74]. These cost-cutting measures result in network congestion that is well known to create "long tail" communication latencies, which severely diminish the performance of interactive and bulk-synchronous parallel applications [75].

The issue of job placement fragmentation is another well-documented challenge in

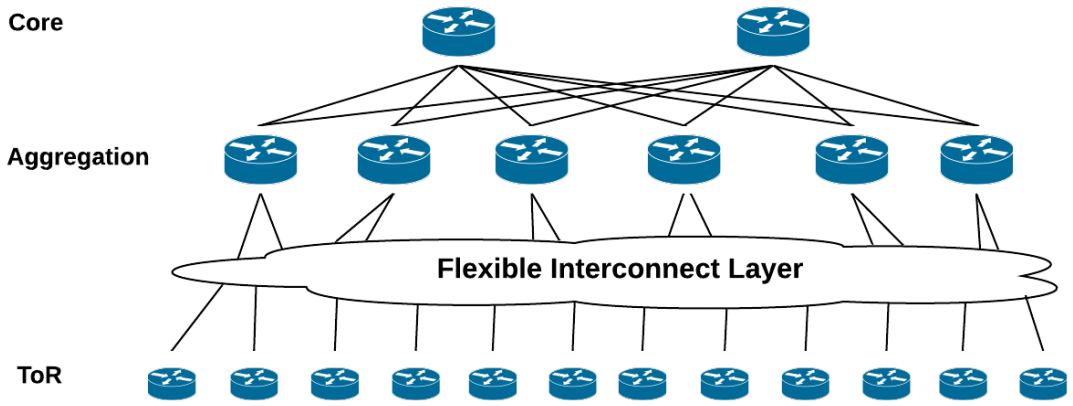


Figure 2.4. Bandwidth steering through SiP switch integration between the ToR and aggregation layer EPSs in a 3-layer Fat-Tree topology

HPC systems that creates detriments in Fat-Tree topologies [76]–[79]. This refers to the phenomenon that arises from multiple applications operating on the HPC system, which are initiating and terminating continuously at different points in time. As a result, applications often receive processor allocations that are physically distant from each other. This means that even the nodes are assigned consecutive message processing interface (MPI) ranks, the nodes themselves are not neighboring and communications between these ranks will require traveling through multiple hops. Figure 2.3 illustrates that in NERSC's Cori fragmentation is both persistent and dynamic. A possible solution to this issue is to migrate the tasks to physically contiguous endpoints, but such migrations take on the order of seconds to complete [80].

The integration of SiP switches allow us to reconfigure the Fat-Tree topology by enabling dynamic controlled bandwidth tapering without introducing additional latency [81], [82]. This is done by using the SiP switches to migrate connections between the ToR and aggregation layer EPSs, which is the lower layer of the Fat-Tree, illustrated in Figure 2.4. By steering the bandwidth at this layer, we can minimize the use of the higher

layers (links between the aggregation and core level EPSs), which allows the higher layers to be bandwidth tapered more aggressively. The overall concept of this method is to reconstruct the locality that was lost due to system fragmentation. Our network will both reduce the cost through bandwidth tapering as well as be less affected by task fragmentation than the static baseline Fat-Tree network with no bandwidth steering. Additionally similar to the Dragonfly topology, bandwidth steering in the Fat-Tree topology also improves the throughput and latency of the network. The throughput is improved through the optimization of bandwith allocation, and the latency is reduced due to packets being able to directly travel between pods (groups of ToR and aggregation EPSs) without having to travel up the tree to the core switches before traveling down again.

Chapter 3

Physical Testbed

3.1 Hardware Setup

Various testbeds were built to demonstrate the feasibility of integrating SiP switches within a conventional HPC environment and to evaluate its performance metrics. Testbeds incorporating both Mach-Zehnder type and microring resonator type switches have been constructed. Additionally, both the Dragonfly topology and Fat-Tree topology have been used with different advantages shown from the integration of the SiP switches, which will be described in more detail in the following subsections. A snapshot of the testbed is shown in Figure 3.1.

In this chapter the different software and hardware components of the testbeds will be described, as well as how they interface and integrate with each other. Additionally this section will also describe the network topologies that were used for experimental evaluation.

Electronic Hardware

For each of the different versions of the testbeds mentioned, we use 16 physical servers equipped with two NetXtreme II BCM57810 10 Gigabit Ethernet Network Interface Cards

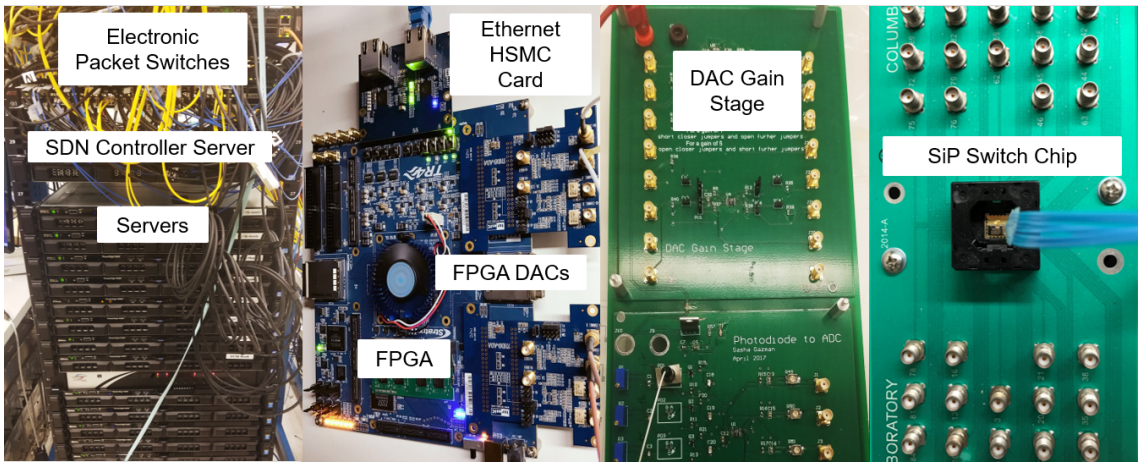


Figure 3.1. Snapshot of the testbed hardware, including the FPGA for controlling the SiP switch, DAC gain stage for amplifying electronic bias signals, EPSs and servers nodes in a rack, and the SiP switch

(NICs), with Intel Xeon 6-core processor and 24 GB of RAM, running Ubuntu. The servers are connected to two top-of-rack (ToR) PICA8 EPSs, which are OpenFlow compatible that allows it to communicate with the SDN controller application. Using the 1G out-of-band campus Ethernet network, the SDN controller sends OpenFlow messages to the EPSs for traffic monitoring and flow table management. Both EPSs have forty-four 10G ports and four 40G ports. Depending on the various topologies used, the EPSs are virtually partitioned into smaller switches. The intra-cluster/pod links are connected with 10G Direct-Attached copper cables, while the inter-group links use SFP+ DWDM transceivers with 24 dB power budget transmitting wavelengths in the C band ranging from C28 to C38 (1554.94 nm to 1546.92 nm). The SDN controller used is Ryu, written in Python [83].

Testbed Topologies

Figure 3.2 shows the system testbed in a Dragonfly topology consisting of 4 groups of 4 EPSs each, with 2 nodes connected to each EPS, for a total of 32 nodes. Each node is a

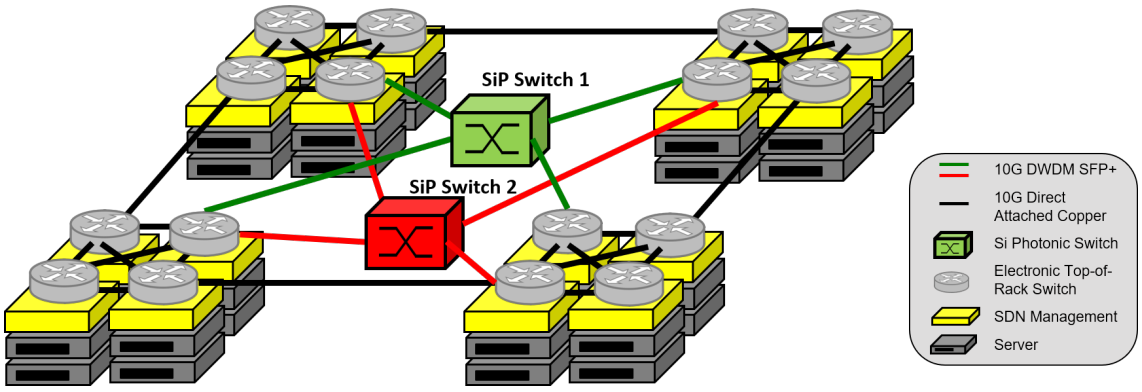


Figure 3.2. Testbed in a Dragonfly topology with two SiP switches for inter-group reconfiguration

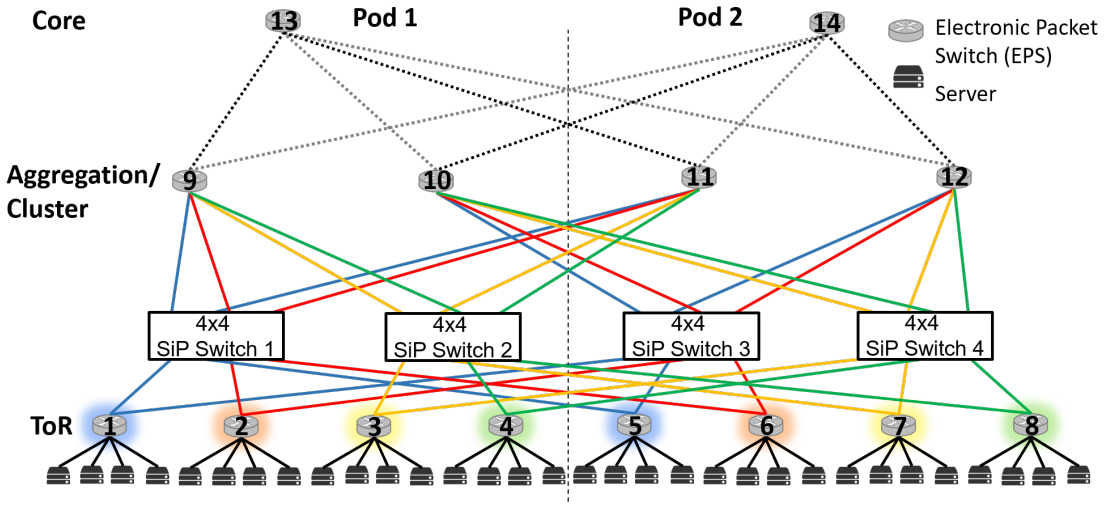


Figure 3.3. Testbed in a Fat-Tree topology with four SiP switches. Here the SiP switches are configured to allow packets from one pod to travel to another without needing to travel to the core level EPSs first, thereby keeping traffic within the lower layer and thus allowing for direct connections for reduced latency and more aggressive bandwidth tapering in the upper layers

virtual machine (VM) on a physical server, with each server hosting 2 VMs. There are up to two SiP switches that connects the four groups of the DragonFly through the EPSs with circulators between each connection. Both MRR and MZI type SiP switches have been used.

Figure 3.3 shows the testbed arranged in a standard three-layer Fat-Tree topology with $k = 2$ (two lower-level switches connect to a higher-level switch). It is also divided

into two pods with 16 nodes per pod, making a total of again, 32 nodes using VMs. Four nodes are connected to each top-of-rack (ToR) packet switch. Each ToR switch has two uplinks to a SiP switch. Therefore, each ToR EPS can have both of its uplinks connect to the EPSs of its own pod, or directly to pod EPSs in the other pod. This allows for flexibility in the network depending on the traffic communication pattern. If the inter-pod traffic is dominant, then the network can be configured to have both uplinks connect to the EPS in the other pod. Of course it also has the capability to be configured as a standard Dragonfly if needed. Note that only uplink fibers are connected to SiP OCSs, not downlink fibers. In this testbed, two of the SiP switches are physical, and the other two are emulated by manually connecting different fibers.

Photonic Switch Integration

Figure 3.4 shows a detailed view of how the servers and EPSs are connected to an MRR switch with four rings, for both the Dragonfly or Fat-Tree topology described previously. Similar to Figure 1.2 we can observe the control plane and data planes, with the SDN controller using OpenFlow to communicate with the EPSs on the left, and using the FPGA Controller to communicate with FPGAs that send bias signals to the SiP switch. In this specific case, as the MRR switch is a wavelength selective switch, each virtual EPS acts as the ToR for a specific rack of servers, and each EPS transmits a specific wavelength - C26 (1556.55 nm), C28 (1554.94 nm), C34 (1550.12 nm), and C38 (1546.92 nm). Each wavelength is multiplexed together and amplified entering the optical bus waveguide of the SiP switch. Depending on how the microrings are tuned with the input bias signals,

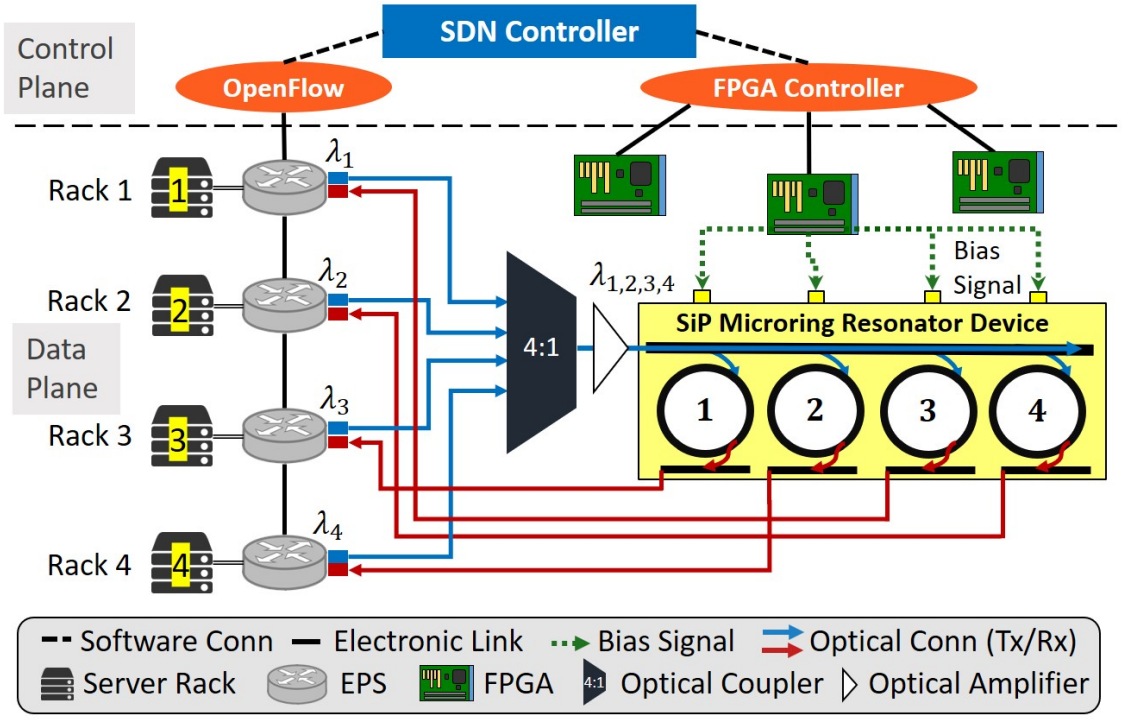


Figure 3.4. Interfaces between the control plane and data plane components. It also shows how the 4 racks of servers are connected to the SiP switch, in this case a 4-ringed MRR device

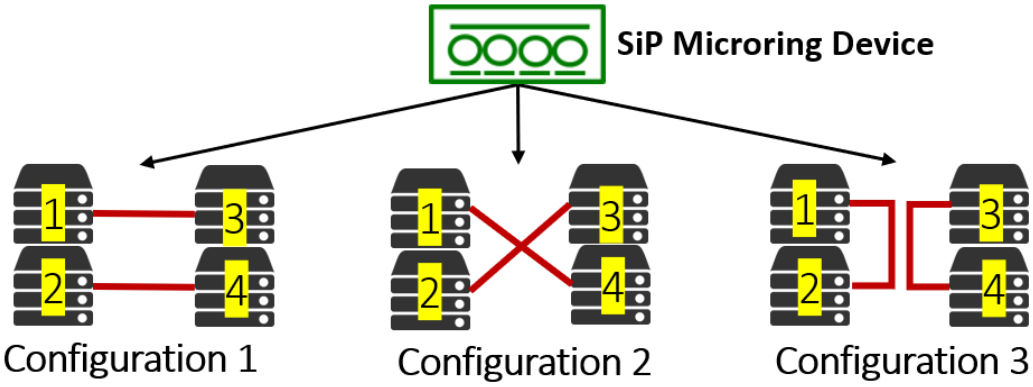


Figure 3.5. Each 4-ringed SiP microring device is capable of these three switching configurations

different wavelengths can be received on a chosen ring. Each ring's output connects back to the receiving end of the optical transceiver that is connected to the EPS. Figure 3.5 shows the possible connections provided by a four-ringed MRR switch for bidirectional connections between four racks of servers.

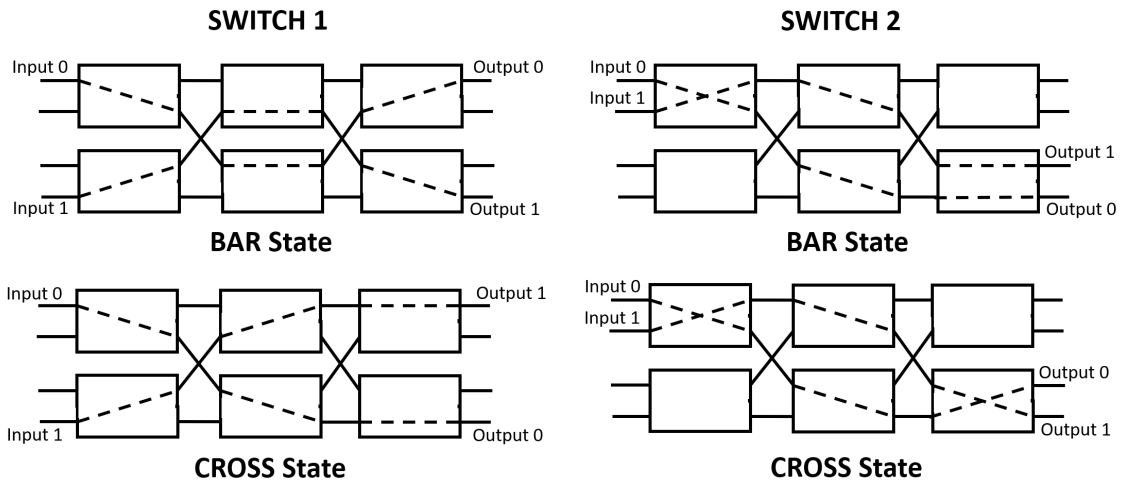


Figure 3.6. Bar and cross configurations for two different MZI SiP switches

We have also used MZI based SiP switches to steer bandwidth between Dragonfly groups, shown in Figure 3.2. Two MZI SiP switches can connect the four groups of the Dragonfly topology through the EPSs with circulators between each connection to create the same flexible topology as MRR based SiP switches described previously. In their default bar state, Switch 1 provides improved connectivity between the left and right halves of the network, while Switch 2 provides improved connectivity between the top and bottom halves. The SiP switches are re-arrangeably non-blocking MZI based 4×4 Beneš topology, with 6 individual MZI elements in each switch. For this work, both switches performs as a 2×2, biased to either bar or cross states. The biasing for these configurations are shown in Figure 3.6. Extinction ratios for both switches range between 10–15 dB.

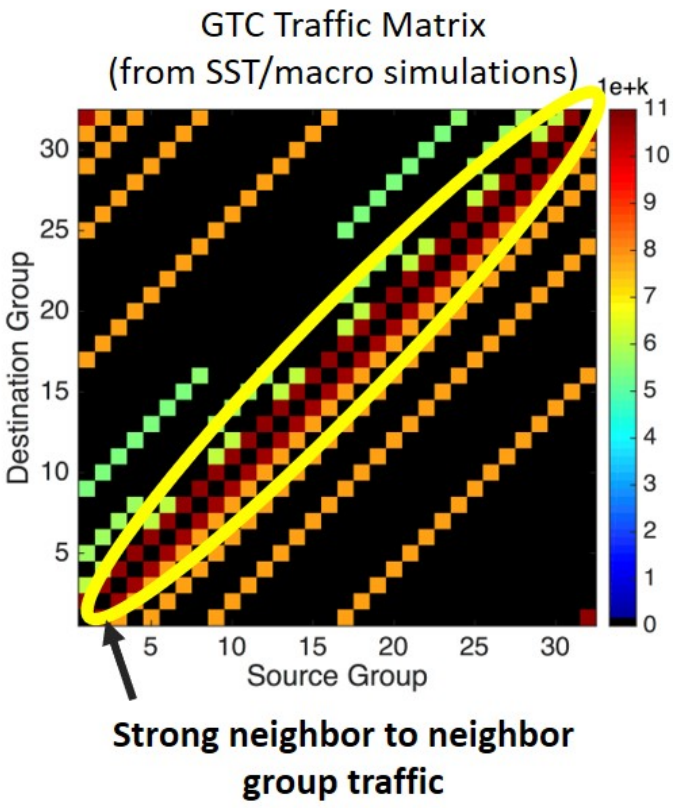


Figure 3.7. Traffic matrix of the GTC benchmark application, showing traffic intensity between different source and destination groups (in a simulated standard Dragonfly topology on SST/macro) [9]

3.2 High-Performance Computing Benchmark Configuration

The testbed uses the Message Passing Interface (MPI) protocol for communication and synchronization of rank assignments over the physical machines. Specifically, MPICH [84] was used. MPI distributes partitions of a given job (ranks) over a network of physical machines, as well as facilitates the coordination of communication between these machines. MPI requires the user to provide the number of ranks to split the job into, as well as a *machinefile*, which contains a list of the IP addresses of the physical machines that

the ranks will be distributed to. Finally, each machine will have a copy of the executable file that is the application. After all of this has been setup, the user can run the HPC benchmark application over a distributed system by arbitrarily choosing a master node and running the MPI command, providing the machine file, number of ranks, and executable file name, upon which the slave nodes outlined by the machinefile will being to work on their given ranks. By default, MPI uses a round-robin approach to distributing ranks among the IP addresses listed in the *machinefile*, which allows us to control how jobs are distributed over the network and thereby the traffic. In a real deployment situation, this would not be possible; however, for experiments, there is occasionally the need to purposefully congest links in order to mimic a specific traffic pattern that is causing a congested network, which can be done by entering the IP addresses in the *machinefile* in a specific order manually.

The application that we operate over the testbed is the Gyrokinetic Toroidal Code (GTC) [63], [85], [86]. As can be seen in Figure 3.7 which shows a heat map of the traffic intensity between different source and destination Dragonfly groups obtained using simulations performed on the SST/macro, the yellow oval highlights the fact that the application displays strong +1/-1 neighbor to neighbor traffic, typical of many other HPC applications. We use a skeletonized version of the GTC benchmark code obtained from the public domain. The process of skeletonization is to remove computation routines while keeping the communication characteristics (packet sizes, destinations, and timing) the same. An example of skeletonization is shown in [87]. This allows for the application to run faster but maintain the same communication pattern (packet sizes, timing, etc.), which translates to a greater bandwidth demand over time. Greater bandwidth demand

was necessary to create congestion in some of the links, in order to show the benefits in performance from relieving this congestion through bandwidth steering with the SiP switches.

Chapter 4

High-Performance Computing Testbed Evaluation

In this chapter we will present various experimental results relating to different aspects of the HPC testbed. This includes topology-agnostic evaluations including the control system latency where each control mechanism latency is shown, and culminating in the end-to-end switching latency [49]. Then we show the system improvements due to bandwidth steering for the Dragonfly and Fat-Tree topologies, by operating the GTC HPC benchmark application on the testbed and comparing their performance between the standard "vanilla" topology to the bandwidth-steered topology [10], [88], [89]. The performance is evaluated on the basis of the application execution time. A faster execution time of the application means that more work can be done on the system, or equivalently more energy saved due to the application finishing earlier.

4.1 Control Plane and Hardware Latency Evaluation

The control plane and its connection to the relevant control hardware was evaluated by measuring the end-to-end latency experienced by packets during a synchronized circuit and packet switching operation. This latency is a combination of software control plane based latencies, and hardware latencies from the EPS and SiP switching time. A visual depicting these latencies is shown in Figure 4.1, and listed in Table 4.1 as well. First, the

Figure 4.1. Timeline showing control plane and hardware latencies

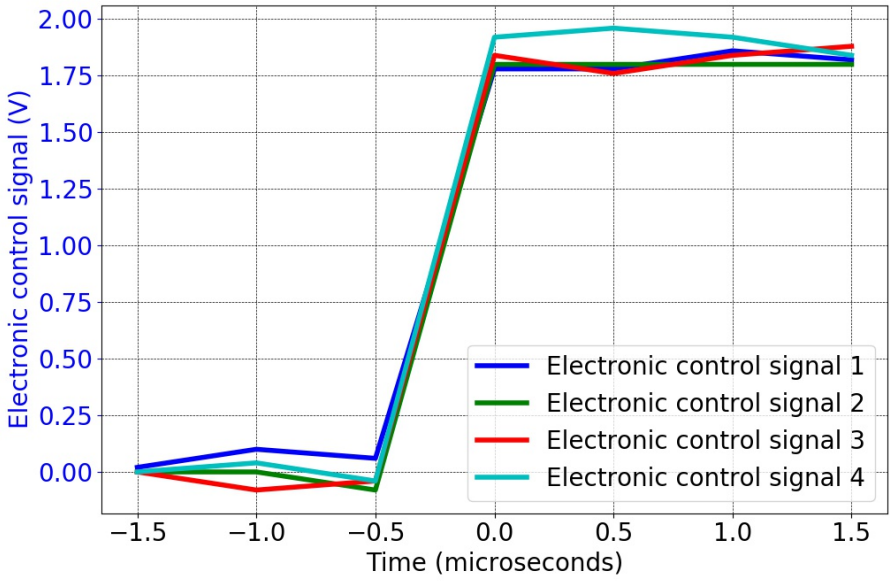
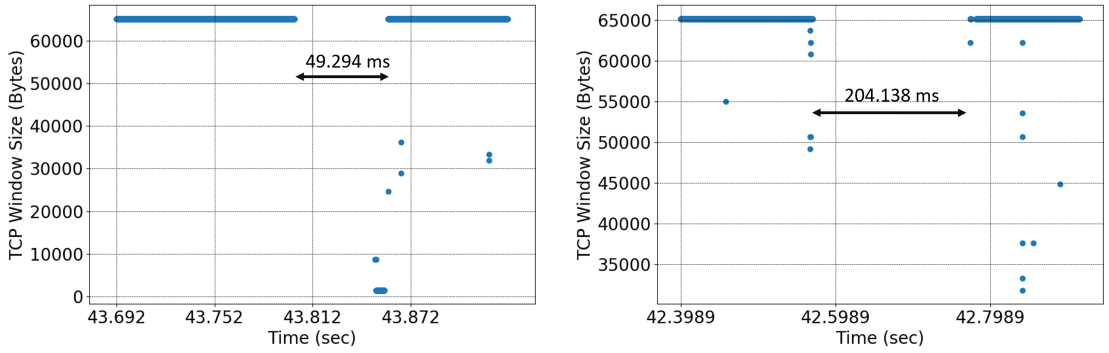


Figure 4.2. Real-time delay between four electronic control signals sent simultaneously

control plane latencies consist of 1) the flow insertion latency, 2) the time for the SDN controller to send a SiP flow update message to the FPGA Controller, 3) the time taken for FPGA Controller to generate an Ethernet packet and send it to the FPGA microcontroller, and 4) the time for the FPGA microcontroller to process the packet and apply the voltage to the DACs. The flow insertion latency from the Ryu SDN controller to a Pica8 switch was evaluated by sending 800 flows in parallel and an average of 78.5 µs per flow was observed. Next, 223 µs was measured for one-half of the Round Trip Time (RTT) of the TCP socket connection that was used for the SDN controller to send a flow update message to the FPGA Controller. This delay occurs in parallel with the EPS flow update latency. The FPGA Controller then takes 702 ns to form the Ethernet packet and send it to the FPGA microcontroller. The latency required for the FPGA microcontroller to process the packet and apply the voltages was measured to be 120 µs, which was found by viewing the Signal



(a) (b)

Figure 4.3. (a) Flow insertion time on an OpenFlow-enabled EPS (Layer 3 switching time), (b) total end-to-end switching time (a majority of which is due to the switch polling time)

Tap logic analyzer and finding the difference between the moment that the flow update packet was received to the moment that the voltages were written to the DACs. The delay between sending multiple electronic control signals simultaneously in software was also measured. Figure 4.2 shows that the four electronic control signals overlap and the delay between them is negligible in real-time.

The hardware latencies consist of 1) the layer 3 switching time of the Pica8 EPSs, and 2) the SiP switching time. The layer 3 switching time of the EPS was measured by performing a data transfer between Servers 2 and 3 on an indirect path, and changing it to a direct path with the same number of hops, and tracking the number of dropped packets with the program *tcpdump*, shown in Figure 4.3(a) [90]. In this figure, each dot represents a single packet in time. The output of tcpdump shows the timestamp of each packet that was sent which is graphed in the x-axis, and the window size at the receiver side over this transmission in the y-axis. The window size itself is not an important metric, but the focus of this graph is to show the gap in the middle of the graph which represents the link unavailability time. For MZI switches, the switching time was measured for both bar to

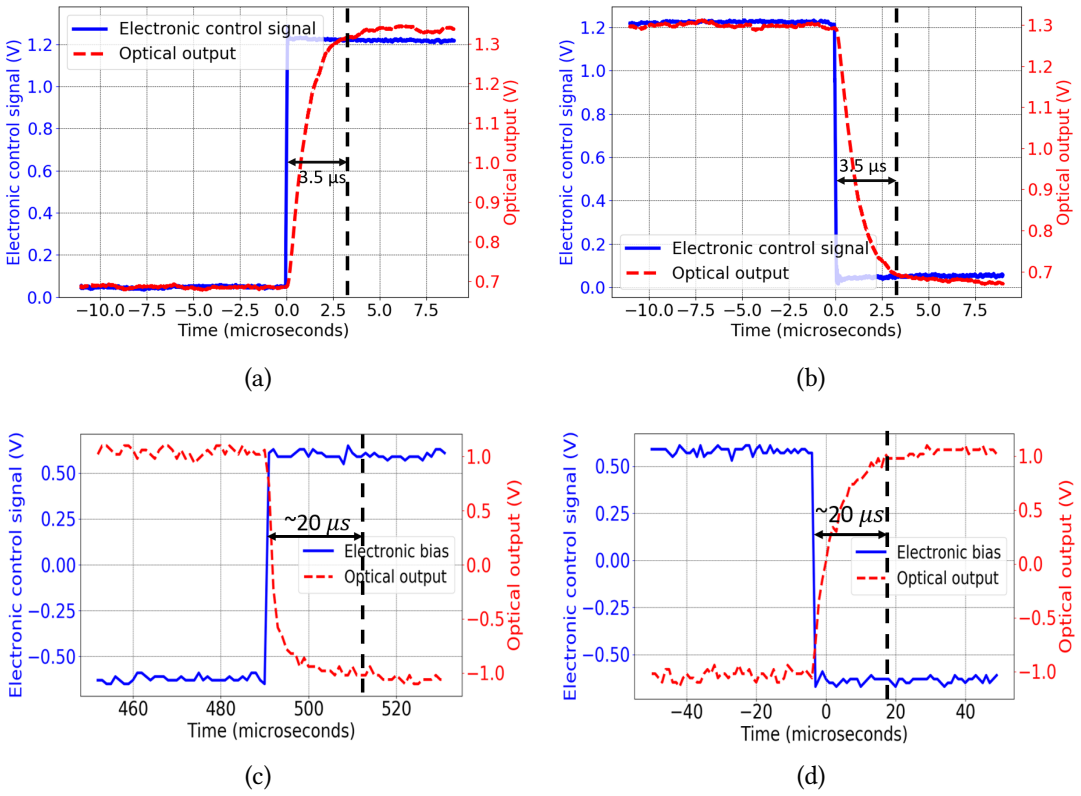


Figure 4.4. ON and OFF switching time for MZI (top) and MRR (bottom) SiP switches

cross and vice versa, which are shown from Figure 4.4(a) and 4.4(b). Each of these were measured to be 3.5 µs. For MRR switches, the switching time was measured to be 20 µs for a given ring to either receive the signal or let the signal pass by the ring (Figure 4.4(c) and 4.4(d)).

We measured the total control plane latency to be 344 µs, which begins from the moment that the SDN controller begins the switching operation by sending flow updates to the EPS and configuration message to the FPGA Controller, to the point where the FPGA microcontroller has applied the voltage to the DACs. Having evaluated all the individual delays of the switching latency, we performed a switching operation on the testbed during a 10 Gbps data transfer and monitored the end-to-end switching latency by observing

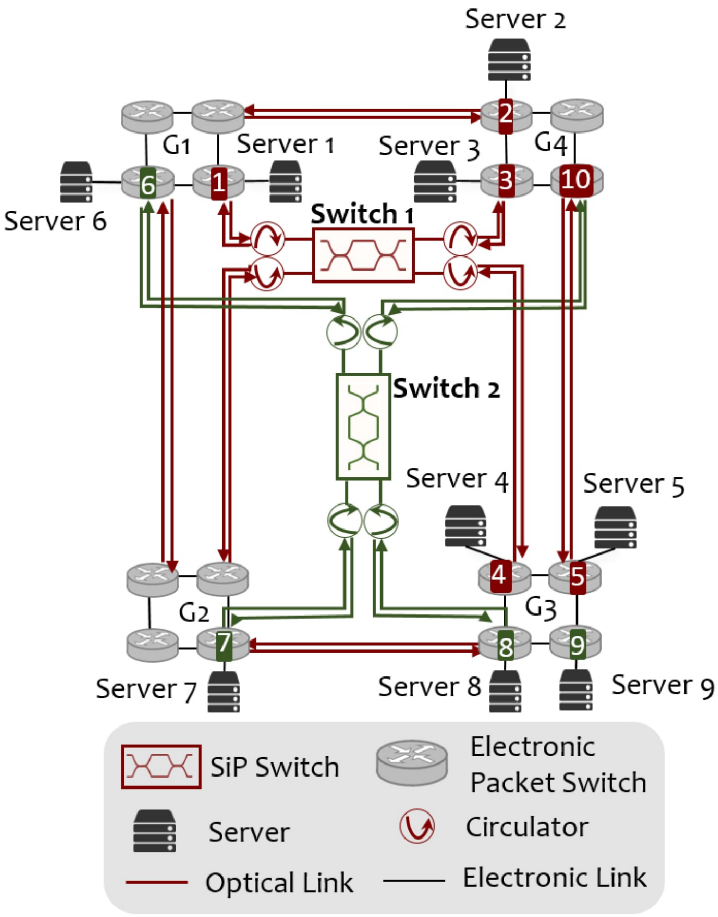


Figure 4.5. Testbed in Dragonfly topology, which is a 2-D version of Figure 3.2 with certain relevant servers and EPS nodes referenced in the experiments labeled

the total packets dropped, using the software tool *iperf* to perform the data transfer [91]. Referring to Figure 4.5, this was done by initially sending data from Server 1 to Server 4 with Switch 1 in the bar configuration, so that the path was an indirect route that passed through EPS 3 before reaching Server 4. Following this, Switch 1 was set to the cross configuration which provides a direct path from Server 1 to 4. Figure 4.3(b) shows the *tcpdump* traces for this operation, and show that no packets were transmitted for a duration of 204.138 ms. The reason for this large delay of an extra 203.794 ms (obtained by subtracting the latency of the control plane from the total end-to-end switching latency)

Table 4.1. List of all software and hardware latencies

Control Plane Latency		Hardware Latency	
SDN controller per flow update	78 µs (parallel 1)	EPS L3 Switching	49 ms (parallel 2)
SDN controller sending SiP flow update message to FPGA Controller	223 µs (parallel 1)	SiP Switching	3.5 µs
FPGA Controller to process message and to send packet to FPGA microcontroller	702 ns		
FPGA microcontroller to process the packet and apply bias signals to FPGA DACs	120 µs		
Total Control Plane		344 µs	
Transceiver locking and switch polling		204 ms (parallel 2)	
Total end-to-end switching		204.3 ms	

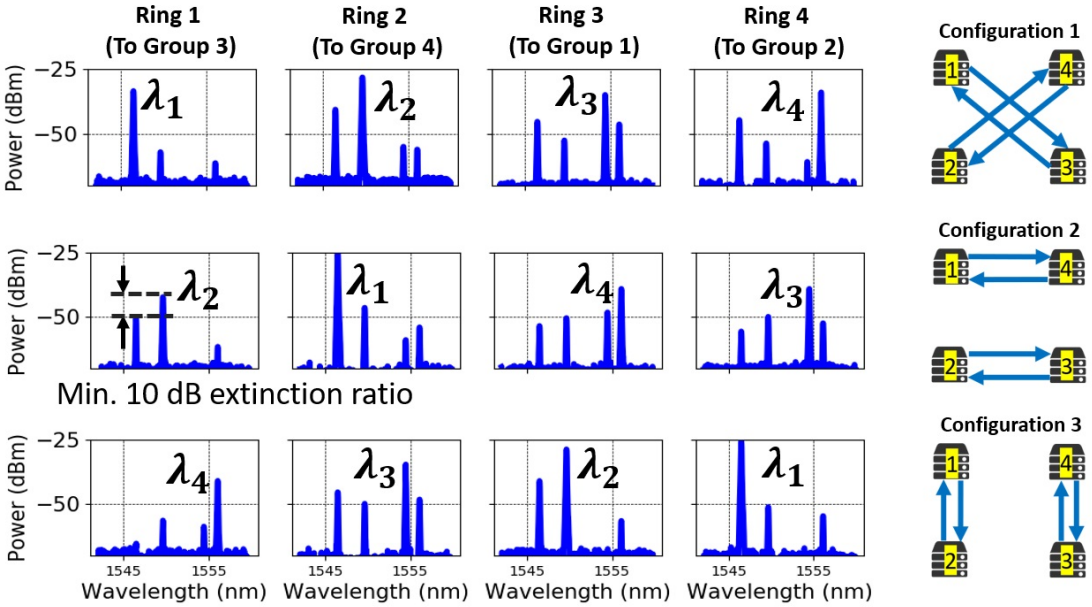


Figure 4.6. Spectra of the four input wavelengths signals seen by the receiver side of the transceivers after being received by the microring resonators of the SiP switch

is due to the transceiver locking and switch polling time, which refers to the time taken by the transceivers at the source and destination ports connected to Server 1 and 4 to recognize each other's signals and to configure their mode of operation over a link, as well as the time for the Pica8 switch to poll the status of the transceivers and to report the link up status that allows the data stream to flow.

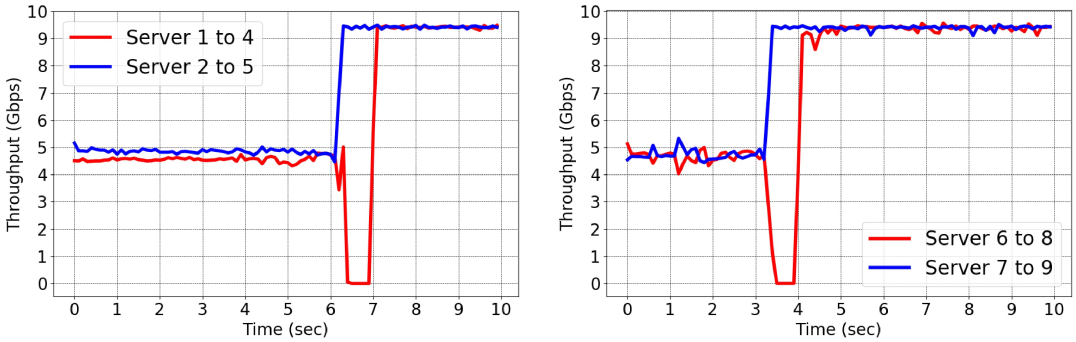
We measured the MRR switch performance under commercial transceiver signals be-

ing passed through it. Figure 4.6 shows the spectra of the four input wavelengths for each switch configuration as seen by the receiver side of the commercial 10G SFP+ transceiver that is plugged into the EPS, without amplification. There are four peaks corresponding to the input signals from each group, and the highest peak is the desired signal that is received by tuning the microring resonator to that wavelength, while the other peaks are crosstalk. For the signal to be received properly, there is a minimum of 10 dB extinction ratio between the two highest peaks. The fiber-to-fiber loss of the SiP switch is approximately 10 to 15 dB, so that an optical amplifier is required to amplify the input signals before entering the SiP device.

Basic Demonstration of Network Congestion Relief

Lastly, to demonstrate the potential benefit in data throughput by relieving congestion over the network by our control plane, we show network optimization through bandwidth steering for each SiP switch individually and in a combined operation. These experiments were performed using the two MZI switches. Because the control plane latency is much higher than the switching latency of the SiP switches, using MRR or MZI switches to perform the following experiments described will not have changed the shape of the output graph.

Figure 4.7(a) shows throughput over time for Servers 1 and 2, which are labeled in Figure 4.5. Initially, Switch 1 is in the bar configuration, and Server 1 is transmitting to Server 4 via the path EPS 3 - EPS 10 - EPS 5 - EPS 4. Meanwhile, Server 2 is transmitting to Server 5 via the path EPS 2 - EPS 3 - EPS 10 - EPS 5. This causes the two data streams



(a) Switch 1

(b) Switch 2

(c) Switch 1 and 2 combined

Figure 4.7. Throughput increase by bandwidth steering to relief congestion

to share the same path from EPS 10 to EPS 5, which causes their throughput to be limited to half the link capacity, or approximately 5 Gbps. After 6 seconds, the state of Switch 1 is changed to cross, which allows Server 1 to send data to Server 4 directly (through EPS 1 and EPS 4), while the route for Server 2 to transmit to Server 5 remains unchanged. Now that these data streams have their own dedicated inter-group link, both senders can transmit at near the full link capacity.

A similar scenario is shown for Figure 4.7(b), with Switch 2 initially in the bar configuration. In this case, Server 6 is transmitting data to Server 8 via EPS 7 and EPS 8, while Server 7 is transmitting to Server 9 via EPS 8 and EPS 9. Therefore they share the

link between EPS 7 and EPS 8, and transmit at a limited throughput of approximately 5 Gbps. By configuring Switch 2 into the cross state, Server 6 has a direct inter-group link to Server 8 and Server 7 has a separate dedicated link to Server 9, allowing both of the data streams to transmit near full link capacity.

Figure 4.7(c) shows the same initial and final configurations as described previously, but with both switches performing a bar-to-cross switching operation simultaneously. We can observe that the throughputs in both cases change from the limited rate to near full link capacity at the same time, which shows that our control plane performs its task of controlling multiple SiP switches along with the associated flow deletions and insertions correctly. The fluctuations in the throughput while the flows are shared in the first few seconds compared to that of the previous figures are purely random and do not have any significance.

Discussion of System Control Plane Latencies

The SiP switch with a 3.5 µs switching time for MZI switches and 20 µs switching time for MRR switches is satisfactory for physical layer reconfiguration with an approximate 50 ms latency for the EPS OpenFlow flow update and 204 ms transceiver locking and switch polling delay. The main challenge discovered in our approach is the transceiver locking and switch polling delay, which dominates overall link unavailability time. Delay times vary for different equipment models and data rates, but typical transceiver locking times range in the tens of microseconds, comparable to the switching latency of the SiP switch, while the switch polling delay for various EPSs are in the hundreds of millisecond range.

Because commercial EPS vendors work with physical wiring of input and output ports to the EPS that do not change, and are not able to manipulate the polling rate in the EPS's operating system, current EPSs are ill-equipped to handle scenarios where devices such as SiP switches that manipulates the optical signal to reconfigure the physical topology thus causing frequent link state up and down changes are used alongside them. This is addressed in Part 3 of this thesis, where we modify the kernel of the end hosts to buffer packets if an optical circuit is not available instead of recklessly transmitting. In addition, the development of commercial EPSs with microsecond polling time will allow for the reduction of the overall end-to-end switching latency to sub-millisecond ranges that is equivalent to a few hundred Kb packet drops on a 10G link for online switching. The development of commercial burst-mode receivers with microsecond transceiver locking time will also reduce the end-to-end latency. For example, [92] has demonstrated nanosecond locking time burst-mode transceivers with data rates up to 25 Gbps. As we move from static topologies to optical networks with more flexibility and dynamic switching, EPS manufacturers will deploy ToR systems with lower link up times that are comparable to the switching times of the SiP devices. In the meantime, fast electronic ToR switching that can operate alongside SiP devices will have to require modification such as the ones described in Part 3 to be made, or using custom packet switch designs.

Despite the millisecond range link unavailability time due to the current EPS polling rate, optical circuit switching can still greatly benefit HPC applications that have well-known traffic characteristics between neighbors of nodes. As it is unlikely that the HPC application require rapid circuit switching due to the fact that traffic characteristics of HPC applications do not vary much over time (which will be shown in the following

System Performance Improvements section), optical circuit switching will be used in a mostly preliminary manner to configure the network topology to optimally benefit the application's traffic matrix before the application's actual runtime, or it can be used to switch between major phases of the application. The average runtime of multiple HPC applications occupying an HPC system are in the range of hours, which means that an end-to-end switching delay in the hundreds of millisecond range is acceptable.

4.2 System Performance Improvements

In this section we will present the system performance improvements enabled through SiP switch enabled bandwidth steering on the testbed. The system is operating the GTC application with MPI as mentioned in the previous section. We constantly monitor the throughput over the links in the network using the SDN controller, which is able to communicate with the OpenFlow-enabled EPSs which automatically keeps track of the byte count of each flow. This allows us to gain insight into the congested or underutilized links in the network, as well as how the traffic distribution over links can change over time. But most importantly, we compare the application execution times between the standard static topology to the bandwidth-steered topology. The percentage result is the ultimate indication of overall system improvement, as all other factors in both runs are the same (identical processing nodes, physical bandwidth, and how the application is operated over the system). The only factor that changes is the network topology.

Table 4.2. Performance increase for various job sizes

Number of Ranks	Execution Time (s)		Performance Increase %
	Standard Dragonfly Topology	Bandwidth Steered Topology	
64	30	20	40%
128	46	30	42%
256	105	70	40%
512	252	186	30%

Dragonfly Topology

First we set the testbed in the standard Dragonfly topology, which is described in Section 2.1. The SiP switches used are either one or two 4-ringed MRR SiP switches as described in Figure 3.4, and whose spectra performance is shown in Figure 4.6.

Performance Improvement with a Single SiP Switch

In the first set of experiments, a single SiP switch is integrated into the testbed, and we ran various job sizes (Table 4.2) of our skeletonized GTC application through different number of ranks over a standard Dragonfly topology and a bandwidth-steered topology enabled by the SiP switch. For the skeletonized GTC application, increasing the number of ranks also increases the job size. To demonstrate the bandwidth steering process in more detail, the assignment of ranks to physical machines was done during runtime to purposely cause congestion between Groups 1 and 2, and between Groups 3 and 4. Figure 4.8 shows the throughput over time of the various inter-group links of the testbed network over the total execution time of the GTC application with 256 ranks. The top plot shows this for the standard Dragonfly topology. It can be seen that the links between Groups 1 and 2, and between Groups 3 and 4 are congested (red and blue), while the links between

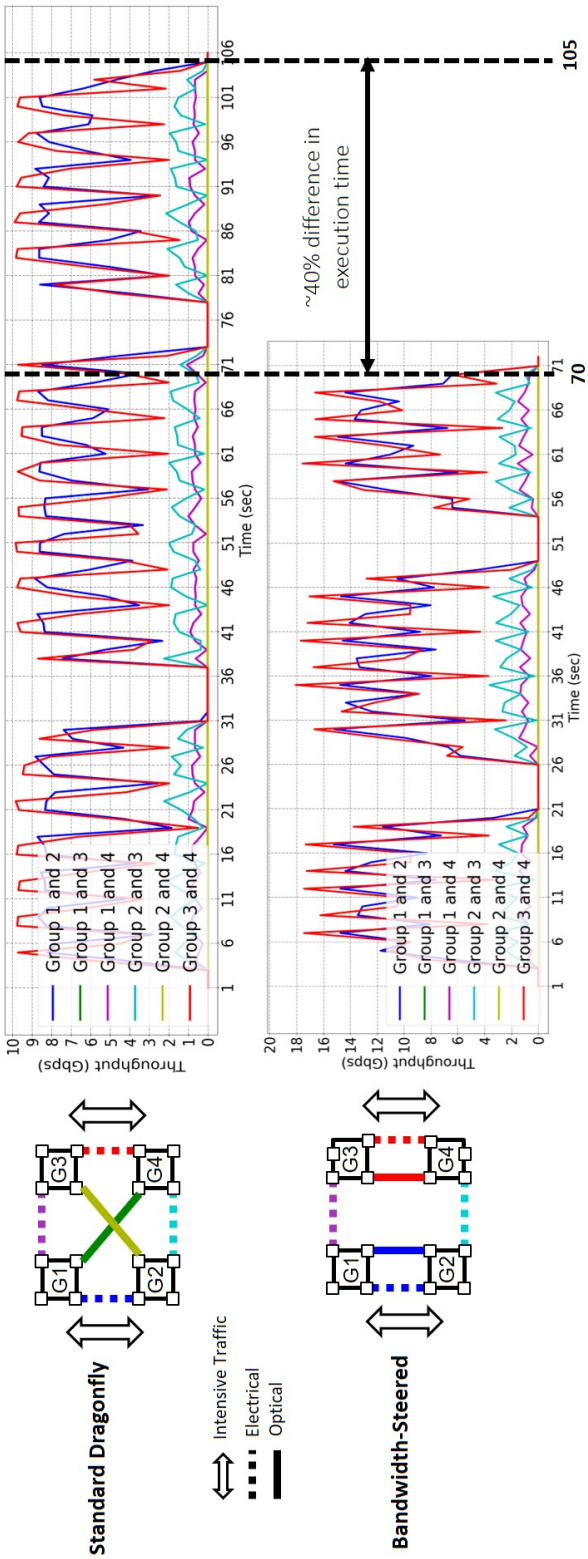


Figure 4.8. Throughput over time comparing a standard Dragonfly topology to a bandwidth-steered topology with a single SiP switch

Groups 1 and Groups 4, as well as between Groups 2 and 3 show approximately zero usage, which matches the rank assignment. The execution time of GTC under this topology is 105 seconds.

Through the identification of the congested as well as the underutilized links from running the application over the standard all-to-all Dragonfly topology, it is clear that the links between Groups 1 to 4 and Groups 2 to 3 should be switched to relieve the congestion between Groups 1 and 2 and Groups 3 and 4, which is done in the bandwidth-steered topology. The result of running the same exact application is plotted on the lower graph. Now with two links each between these pairs of communicating groups, the total bandwidth available between them is increased to a maximum of 20 Gbps, which is reflected in the y-axis of the plot. As can be seen, the traffic between Groups 1 and 2 and Groups 3 and 4 does indeed take advantage of this newly available bandwidth, and its throughput rises to 18 Gbps. We also align the time x-axis of the top and bottom plots and show that the increase in throughput due to bandwidth steering also allowed for the execution time of the application to be reduced by approximately 40%, from 105 seconds to 70 seconds.

Performance Improvement with Two SiP Switches

In the next set of experiments, two 4-ring SiP switches were inserted into the testbed network, in order to demonstrate the increased flexibility of the network to match its topology to the application traffic pattern. For this experiment, rank assignment to physical machines was done in a way to create intensive traffic between immediate neighboring

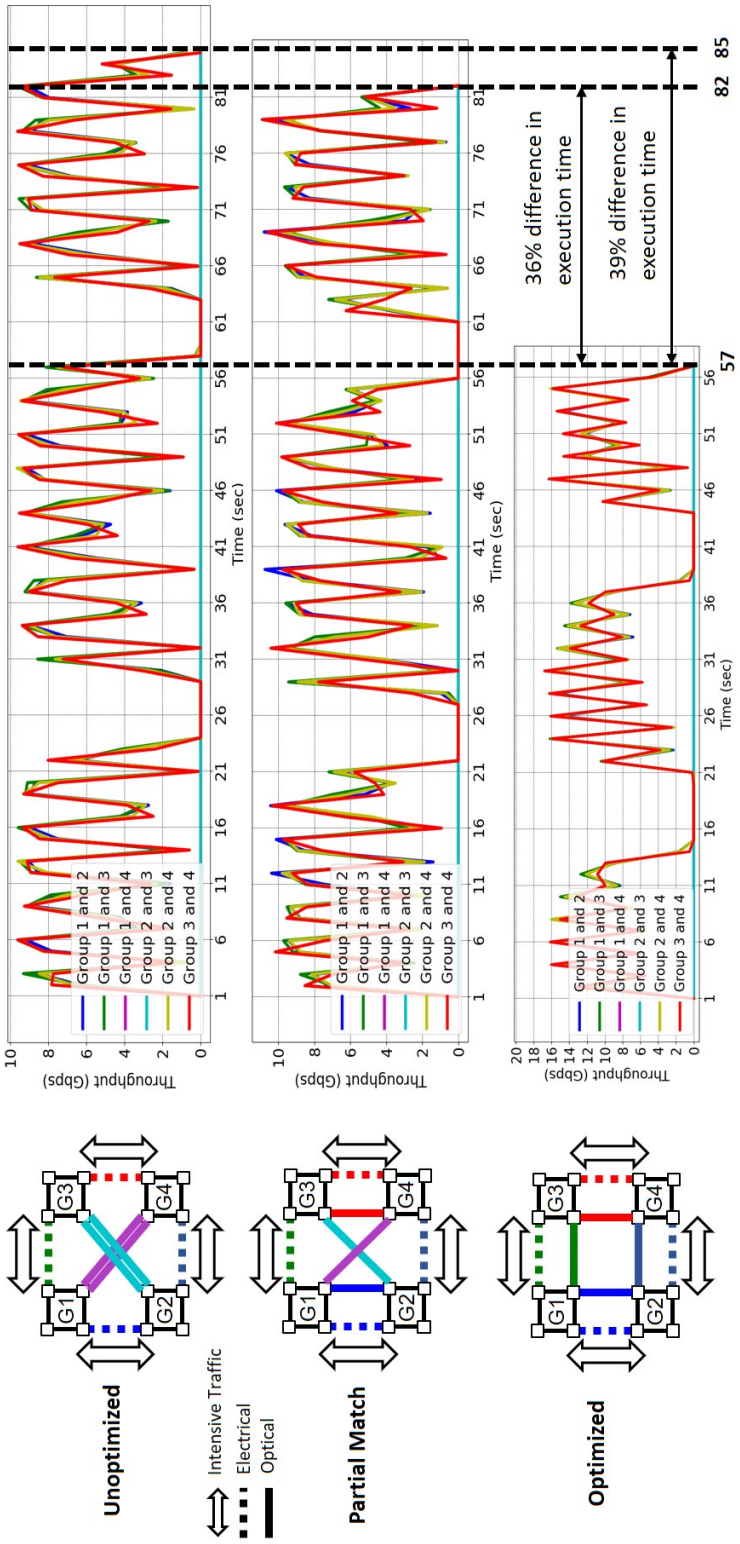


Figure 4.9. Throughput over time comparing different configured topologies enabled through two SiP switches

groups, matching the real GTC traffic matrix shown in Figure 3.7. Under this traffic, we set up three different topologies as shown in Figure 4.9 - *unoptimized, partial match, and optimized* topologies, and run the GTC application to see the difference in system performance that is achieved due to bandwidth steering.

In the *unoptimized* topology, the two switches are both set to cross configuration, which does not help support the traffic pattern that is occurring on the outer edges. It leads to the longest execution time of 85 seconds. In the *partial match* topology, one of the SiP switches is configured to match the traffic, while the other one remains in the cross configuration. This allows for 20 Gbps of bandwidth to be available between Groups 1 and 2, and Groups 3 and 4. Yet despite this newly available bandwidth, the traffic barely takes advantage of it. It can be seen that some spikes in the traffic throughput do rise above 10 Gbps, but it does not use nearly the full capacity available. We theorize that this is due to the 10 Gbps bottleneck that is present between Groups 1 and 3, and Groups 2 and 4. Since computations must wait for nodes communicating between these groups, it slows down traffic in other parts of the system despite higher available network bandwidth. This is reflected in the application execution time of 82 seconds, which is only approximately 3% lower than for the *unoptimized* topology. This is an interesting result as it shows that simply provisioning additional bandwidth between nodes does not necessarily mean that the traffic throughput will increase if there are bottlenecks elsewhere in the network. Instead it is necessary to route the packets to minimize hotspots according to the new topology created by the optical switch. Finally in the *optimized* topology, both switches have been configured to match the traffic pattern, allowing for 20 Gbps bandwidth on all sides. This time there is a clear benefit, and the traffic rises to more than 16 Gbps, and the

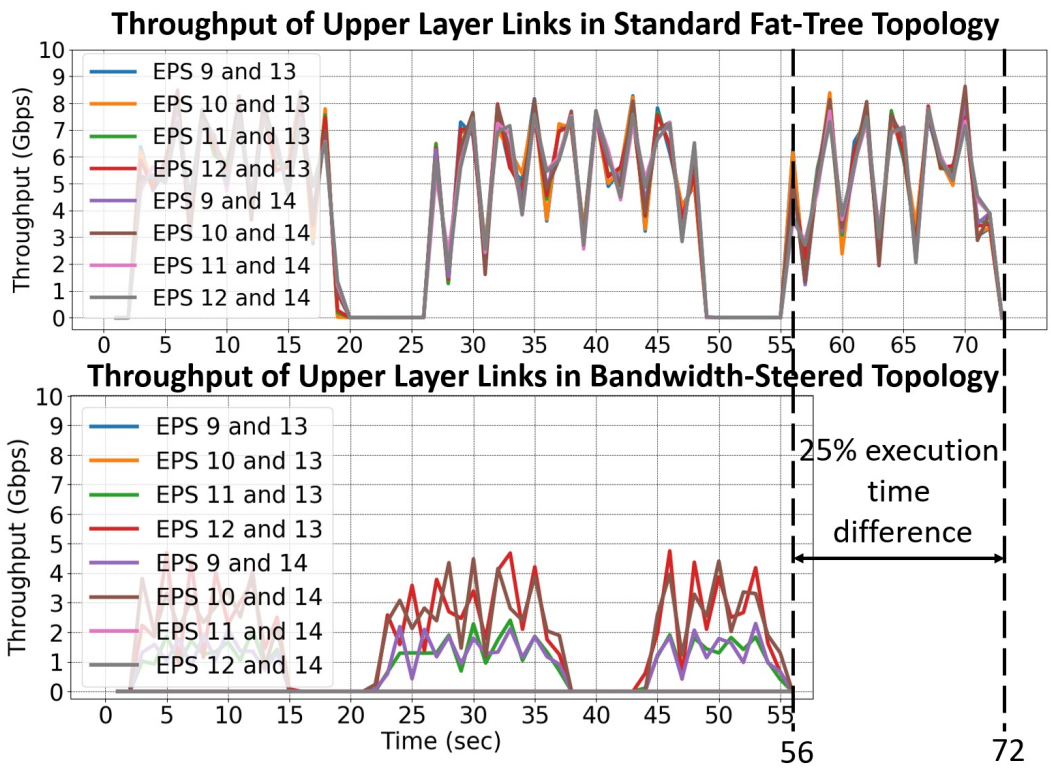


Figure 4.10. Throughput of upper-layer links (between aggregation and core EPSs) over the entire runtime of the GTC application for the standard Fat-Tree topology (top) and the bandwidth-steered Fat-Tree topology (bottom)

execution time is reduced to 57 seconds, a 39% reduction compared to the *unoptimized* topology and a 36% reduction compared to the *partial match* topology.

Fat-Tree Topology

For the following results, the testbed is configured into the Fat-Tree topology as described in Figure 3.3. Once again, two 4-MRR SiP switches are used, and another two of the same types of switches with the same switching capabilities are emulated.

Figure 4.10 plots the throughput of the links in the upper Fat-Tree layer (between the aggregation and core EPSs) over the entire runtime of the GTC application. Once again, each colored line shows the throughput inside one link, identified in the legend through

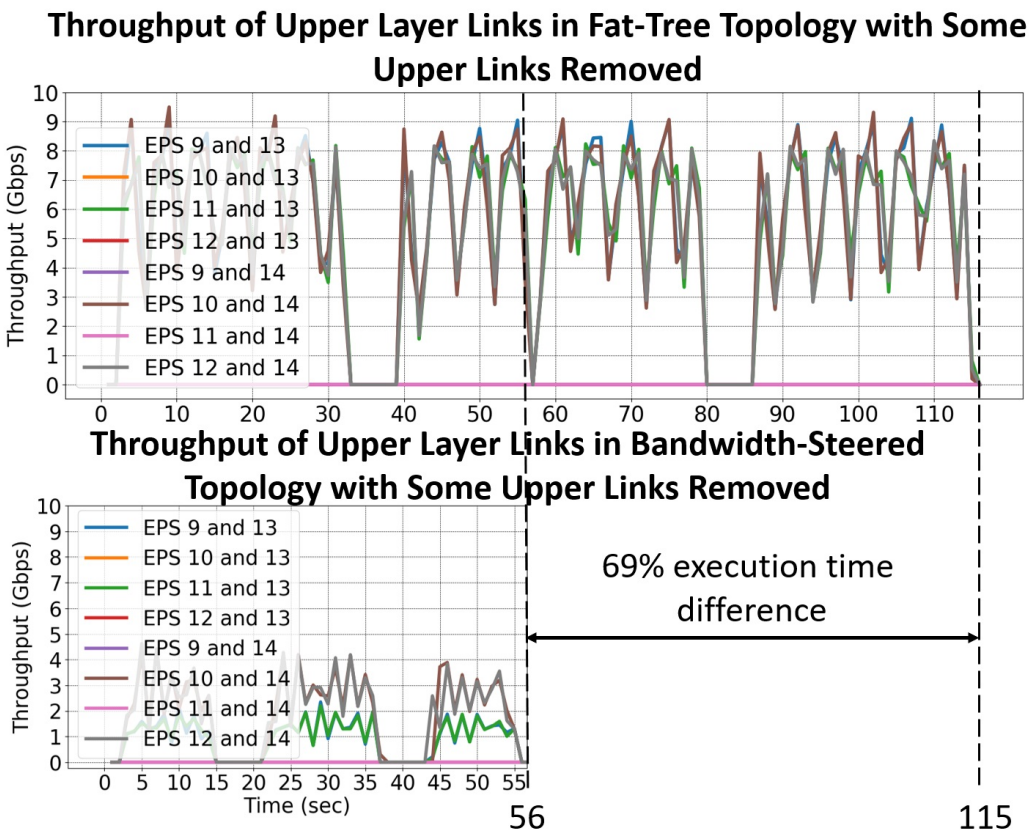


Figure 4.11. Throughput of upper-layer links (between aggregation and core packet switches) over the runtime of the GTC application for the standard Fat-Tree topology (top) and the bandwidth-steered Fat Tree topology (bottom) *with some upper layer links removed* for reducing power consumption

its source and destination EPSs. Traffic intensity in the upper-layer links illustrates the effect of configuring SiP switches to keep traffic in the lower-layer links (between ToR and aggregation EPSs). For the purposes of demonstrating our bandwidth steering concept, we place MPI ranks in a way that maximizes traffic at the top layer of the Fat-Tree. Therefore as can be seen in the top plot of Figure 4.10, when we run the GTC application, each of the upper layer links of the standard Fat-Tree topology are heavily utilized.

In the bandwidth-steered topology, SiP switches are configured in a way that allows both uplinks from each ToR switch to be directly connected to the destination's aggrega-

tion EPS, and therefore bypass the core layer, reducing packet switch hops by two. However, this topology causes any traffic that flows within the same pod to use upper layer links. However, as stated previously, the traffic generated is heavily inter-pod, which means that the bandwidth-steered topology is a better fit. This is reflected in the bottom plot of Figure 4.10, where we observe much lower bandwidth demand in upper-layer links. Only four of the upper layer links have traffic which ranges between 2 to 5 Gbps. The other links are virtually unused, because the bandwidth-steered topology has isolated the traffic to lower-layer links. We also notice that the application executes about 25% faster with the bandwidth-steered topology with 56 seconds, over the standard Fat-Tree topology with 72 seconds.

Following this observation, we can taper the bandwidth of the top-level links by removing links. Thus, we remove the links between EPS 10 and 13, EPS 12 and 13, EPS 9 and 14, and EPS 11 and 14 and perform a similar experiment to the previous paragraph. The results are shown in Figure 4.11. We note that with half its upper links removed, the standard Fat-Tree topology now takes 115 seconds as opposed to 72 seconds to finish running the same application (69% difference). Also we observe that the links that are left are congested, with bandwidth demand reaching over 9 Gbps.

Comparing this to the bandwidth-steered topology of Figure 4.10, we observe that it still takes the same amount of time of 56 seconds to finish, which matches expectations as we observed that the bandwidth steered topology did not use some of the upper layer links in the previous experiment. Therefore, the bandwidth-steered topology can tolerate more tapering with no performance penalty.

It is important to note that while real deployed HPC systems use much higher band-

width links (e.g. 100G InfiniBand), the benefits of the bandwidth steering still apply, regardless of link data rate.

Chapter 5

Conclusion and Discussion

In the first part of this thesis we introduced the concept of bandwidth steering using low-radix silicon photonic switches for HPC systems. The motivation arises from the mismatch in HPC application traffic and the physical distribution of bandwidth resources in the interconnection network, due to the skewness of many HPC applications as well as defragmentation of rank placement to physical machines over time. This results in a suboptimal usage of network resources, lowering performance and increasing energy usage.

We address this problem by inserting low-radix silicon photonic switches into the HPC network, so that the network becomes flexible and can reconfigure its physical topology to the network traffic. The integration of these photonic switches into the system was described in Chapter 1, where we used an SDN control plane to coordinate the states of both the photonic switches and the top-of-rack electronic packet switches to realize intelligent adaptation to traffic. We targeted the Dragonfly and Fat-Tree network topologies (Chapter 2, which are widely used topologies in deployed HPC systems. A physical testbed was built for both types of topologies and different types of photonic switches (Chapter 3, and we evaluated each of these in Chapter 4 for feasibility and performance compared to standard static topologies.

Through the experimental demonstrations presented in Chapter 4, we showed that the bandwidth steering concept is feasible for a conventional electronic system and can be realized using the low-radix SiP switches, with performance improvements of 40% for Dragonfly and 69% for Fat-Tree topologies. The insertion of SiP switches in Fat-Tree topologies also allow for bandwidth tapering in the Fat-Tree without any detriments to performance. Through the demonstrations of these experiments, we can conclude that not only can the execution time of the system be significantly affected by the bandwidth available between each processing node, but that it is also proportional to the amount of throughput increase that is gained through bandwidth steering.

One may question the of use of SiP switches over additional electronic packet switch for bandwidth steering. The primary motivation for using SiP switches over an EPS is the fact that the SiP switch can not only take advantage of the WDM capability of optical communications to carry a large amount of channels from electronic router groups, but also switch them in a simple manner that requires little changes on the software side because it occurs purely on the physical layer. The network architecture features a flat topology where the SiP switches are layer 1 devices that are invisible to the electronic system, and the only part that needs to be potentially changed is the flow rules on the EPSs as it dictates routing. In terms of using SiP switches over other types of optical switches such as MEMS, SiP switches are limited in their port count but have much lower energy consumption, cost, and footprint. However their main advantage over MEMS switches is their ability to be closely integrated with electronic control drivers and other peripherals.

5.1 Future Work

Future work related to the topic of bandwidth steering using silicon photonic switches can be expanded on many fronts. The methodology of photonic switch integration with conventional datacom environments can be improved with lower latecy by using specialized application-specific integrated circuits (ASICs) that can be interfaced with the SDN controller. The ASICs can communicate with the SDN controller using OpenFlow just like the electronic packet switches do, so that the SDN controller will use to the same language to communicate with all switching devices, photonic or electronic, making operations simpler for the user.

In terms of applications, the concept of bandwidth steering is not limited to HPC, but can be applied to *any* application, such as data center applications and machine-learning applications. However, the application traffic characteristics must be suitable for bandwidth steering to provide performance improvement in all of these applications. This means that the traffic is skewed, so that some links are congested while others are underutilized at the same time. This allows for the uncongested links to be ”moved” by switching the silicon photonic switches to relieve congested links. If the traffic is evenly distributed over the system, then switching links around will not improve performance. Second, the volatility of the traffic pattern (how frequent the traffic pattern changes over the system) relative to the end-to-end switching speed must be taken into account. In essence, the benefits of switching to relieve congestion must outweigh the detriment of dropping packets during switching. This is the main reason why the applications targed in this thesis were HPC applications, as their traffic pattern either never changes over the

entire runtime of the application, or only during major phases of the application, which is on the order of hours, while the end-to-end switching time was hundreds of milliseconds.

Part II

Autonomous Network and IT Resource Management for Geographically Distributed Data Centers

Chapter 6

Network Architecture and Control Strategy

6.1 Introduction

Data centers (DCs) have become essential infrastructure for commercial businesses across all industries, with increasing demand for connectivity due to emerging applications continuously driving the need for robust, high bandwidth and low latency data interconnect solutions. Cisco predicts a sevenfold increase in mobile data traffic and 12-fold increase in virtual reality and augmented reality traffic from 2017 to 2022, as well as other applications such as Internet video to TV, video surveillance traffic, gaming, coinciding with the advent of 5G and the Internet of Things. Metro data centers will have to carry a third of the total global IP traffic that is expected to triple to 4.8 zettabytes by 2022 from 1.5 zettabytes during 2017 [25].

Driven by this traffic demand as well as the growing trend of using Content Distribution Networks (CDNs) by an increasing number of businesses to deliver content to users with much lower latency, enterprises are moving towards deploying a vast number of small- to mid-sized DCs separated by metro-scale distances connected with a fiber network [27], [93]–[95]. Because DC networks form the groundwork that supports this growth, contention and low utilization of shared network and compute resources in DC

networks can be a major contributor to performance degradation. This is especially true for cloud computing services built on a virtualized infrastructure, where much of the compute resources are shared among multiple end-users and can show significant load fluctuations due to user demand. Specifically, DC workload patterns have been found to have weekend/weekday variations [96] but exhibit burstiness and are generally unpredictable on shorter time scales [97], [98]. To maximize utilization of the underlying physical infrastructure for unpredictable traffic behavior, an adaptive DC resource management strategy that provisions resources on-demand is required [99], [100]. There are two types of targeted resources: network resources, primarily link bandwidth, and IT resources, including CPU, physical memory, and hard disk capacity. Network resources can be effectively managed at the flow level over the DC network, and IT resources are managed through allocation and migration of virtual machines (VMs) in a cloud data center.

The distribution and management of network and IT resources is a task suited to Software-defined networking (SDN) [101], [102]. An SDN controller essentially collects network resource information and disseminates control plane intelligence to the physical network entities to adapt the network to current load conditions. This is enabled by the separation of the data and control planes which allows centralized management of network components from the higher layers. We have shown in the first part of this thesis that the SDN controller is capable of regular monitoring of network domain traffic behavior as well as the ability to mitigate network congestion by intelligently allocating dynamic bandwidth resources to heavily subscribed links. Additional modifications to the SDN application allows the controller to obtain IT resource usage statistics of the

servers in the DC and subsequently control Virtual Machine (VM) placement to achieve high server utilization. The combination of these capabilities allows the SDN controller to perform IT resource management with awareness of the bandwidth usage of the compute elements and links in the network, creating an additional degree of control for network-wide performance optimization.

In this part of the thesis, we show an autonomous and on-demand network and IT resource provisioning SDN control plane built for an optically converged network architecture for metro-scale data center networks [94], [103]. A converged network architecture can setup dynamic lightpaths between racks in different data centers. This guarantees lower delays for transferring data between DCs with respect to conventional metro and DC networks. However, our network architecture combines both network and IT resource provisioning capabilities to support specific Quality of Service (QoS) requirements and optimizes resource utilization under rapid and dynamically changing load variations. At the same time, it aims to minimize cost and energy usage of the distributed DC computing elements by consolidating server workloads. This is achieved through i) second-scale monitoring of both the link bandwidth usage of the network and IT resources usage of the racks, ii) enabling connections with various QoS through two types of links, namely background and dynamic, iii) triggering autonomous live-migrations of VMs operating in the servers to consolidate workload, and iv) managing network connections in both explicit operations (request by operator) and implicit operations (automated assignment based on traffic characteristics) by the SDN control plane. An experimental testbed of three data center nodes, two fully emulated and one in the control plane, was built to demonstrate the capability to provision both network and IT resources under several scenarios.

While there have been many related works investigating dynamic provisioning of bandwidth in DC networks or server-resource aware VM consolidation for managing IT resources, there has not yet been any that combines the allocation of both types of resources. Works that feature VM consolidation use two types: static and dynamic. Static consolidation uses historical resource utilization as input to predict future resource use trends in order to map VMs to physical hosts, whereupon the assignment remain unchanged for months [104]. Dynamic server consolidation is useful for a more unpredictable workload and is carried out on a shorter timescale. Examples of this include [105], [106] and [107]. [105] presents a framework for the placement of VMs to minimize network power reduction by turning off unused ToR switches while satisfying as many network requests as possible.

Network bandwidth is a resource crucial to the performance of the majority of cloud-based services. There are a large number of strategies proposed for fine-grained network resource provisioning for DC networks. [108] describes a flexible-grid inter-data center network architecture and investigates efficient support of cloud and big data applications. [109] presents a flexible optical metro node architecture and discusses methods and challenges in providing dynamic transport services in a distributed data center environment. [110] and [111] demonstrated optical burst/packet switched network architectures capable of setting up direct lightpaths between racks in different metro data centers, which reduces inter-DC latency and provides better resource usage compared to conventional networks.

In these works, different methods for adaptive provisioning IT or network resources are shown, but each focuses on only one type of resource, either IT or network band-

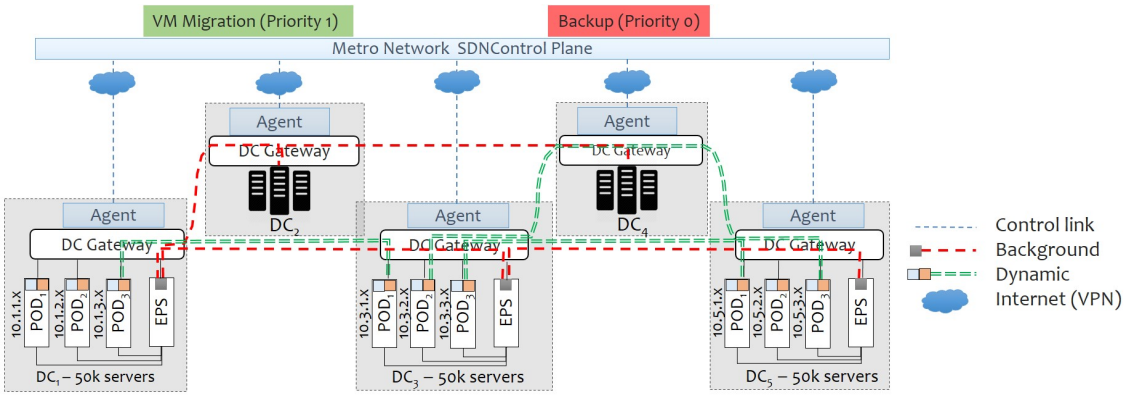


Figure 6.1. Control plane provisioning network and IT resources over a optical converged intra- and inter-data center network, providing quality of service (QoS) differentiation through background and dynamic connections, and consolidation of VMs

width. Works involved in the management of IT resources use the current state of the network traffic to make decisions on provisioning VMs, but do not influence the network's physical topology or network capacity with additional resources. On the other hand, works involved in provisioning dynamic bandwidth do not address the issues of overusing server CPU or memory. We seek to create a comprehensive control plane that manages both types of resources in a synergistic manner.

6.2 Data Plane

The data plane consists of the inter/intra network and the racks/pods of the data centers, shown in Figure 6.1. The inter-DC connectivity is provided by a data center gateway, which can be through the EPS, or using an optical gateway that is a Colorless, Directionless, Contentionless, Reconfigurable Optical Add/Drop Multiplexer (CDC-ROADM). The EPS network provides both the intra-DC connectivity as well. The network architecture supports two types of connections: i) background (shown in red), which provide basic

DC-to-DC connectivity and are present at all times, and ii) dynamic (shown in green), which provides direct rack/pod connections between DCs, which transmits through the DC gateway. In the single-rate scenario, all connections are at the same rate and in multi-rate scenario dynamic connections are at higher rates (40G, 100G, etc.).

The background connections are shared by several traffic flows from different Top-of-Rack (ToR) switches. As these flows share the total bandwidth of the link, their throughput is proportionately limited by the number of flows sharing the link. Therefore, background connections are intended for short-lived and low data rate traffic flows, which require no strict requirement in terms of bandwidth or latency. A set of background connections is always active to provide basic connectivity among all the distributed data centers. On the other hand, dynamic connections are used to perform large data rate transfers between racks in different data centers. They are provisioned on-the-fly directly between racks or pods through the DC gateway and are used for applications with strict latency or large bandwidth requirements. At high network loads, it is possible that a dynamic connection is blocked due to the lack of wavelength resources. When this happens, the setup of the dynamic connection is rescheduled at a later time, which increases the total time required to complete the data transfer. However, as this would negatively affect critical data transfers such as the migration of virtualized 5G network services, we assign a priority to each dynamic connection and we employ a control algorithm for efficiently manage the priority levels. The direct rack-to-rack and/or pod-to-pod connections in this architecture also enables transparent data center to data center connections without Electrical/Optical/Electrical (OEO) conversions. Optical metro-scale transceivers supporting a transparent reach up to 80 km provide the direct connections at low cost.

Algorithm 1 Network and IT resource implicit control algorithm

New connection request **R** from DC *S* to DC *D*;
Monitor the active Background and Dynamic connections, and IT resources usage percentage;
if Traffic on Background **B** DC *S* to DC *D* $\geq$ Threshold $\mathbf{T_0}$. **then**
  if SD of traffic on B ≤ 1 **then**
    Create new Background link from DC *S* to DC *D*
  else if SD of traffic on B > 1 **then**
    Create new Dynamic link from DC *S* to DC *D*
  end if
end if
if IT resources usage on Server **U** $\geq$ Threshold $\mathbf{T_1}$
or Dynamic link bandwidth usage on Server **Q** $\geq$ Threshold $\mathbf{T_2}$ **then**
  Initialize live VM migration from Server **U** to next available Server **V**
else if IT resources usage on Server **V** $\geq$ Threshold $\mathbf{T_3}$ **and** IT resources usage on Server **U** < Threshold $\mathbf{T_4}$
or Dynamic link bandwidth usage on Server **V** $\geq$ Threshold $\mathbf{T_4}$ **and** Dynamic link bandwidth usage on Server **U** < Threshold $\mathbf{T_5}$ **then**
  Initialize live VM migration from Server **V** back to Server **U**
end if

6.3 Control Plane

The SDN control plane (Figure 6.2) contains three modules: i) the traffic and IT resource usage monitor, ii) the resource usage optimizer, and iii) the topology and virtual machine (VM) placement manager. Data center are divided in different subnets based on their size, e.g. DC1 has the 10.1.x, DC2, 10.2.x, etc. ToRs and/or Pods are SDN-compatible switches with permanent flow rules for global inter data center connectivity through background connections. At initial startup, the SDN control plane establishes connections with both the OpenFlow enabled switches and the DC servers. At regular second-scale intervals, the SDN controller obtains current rack-to-rack traffic statistics from the OpenFlow switches, as well as IT resources usage of each server in the DCs. The resource usage optimizer uses this information to determine whether new background or dynamic connections

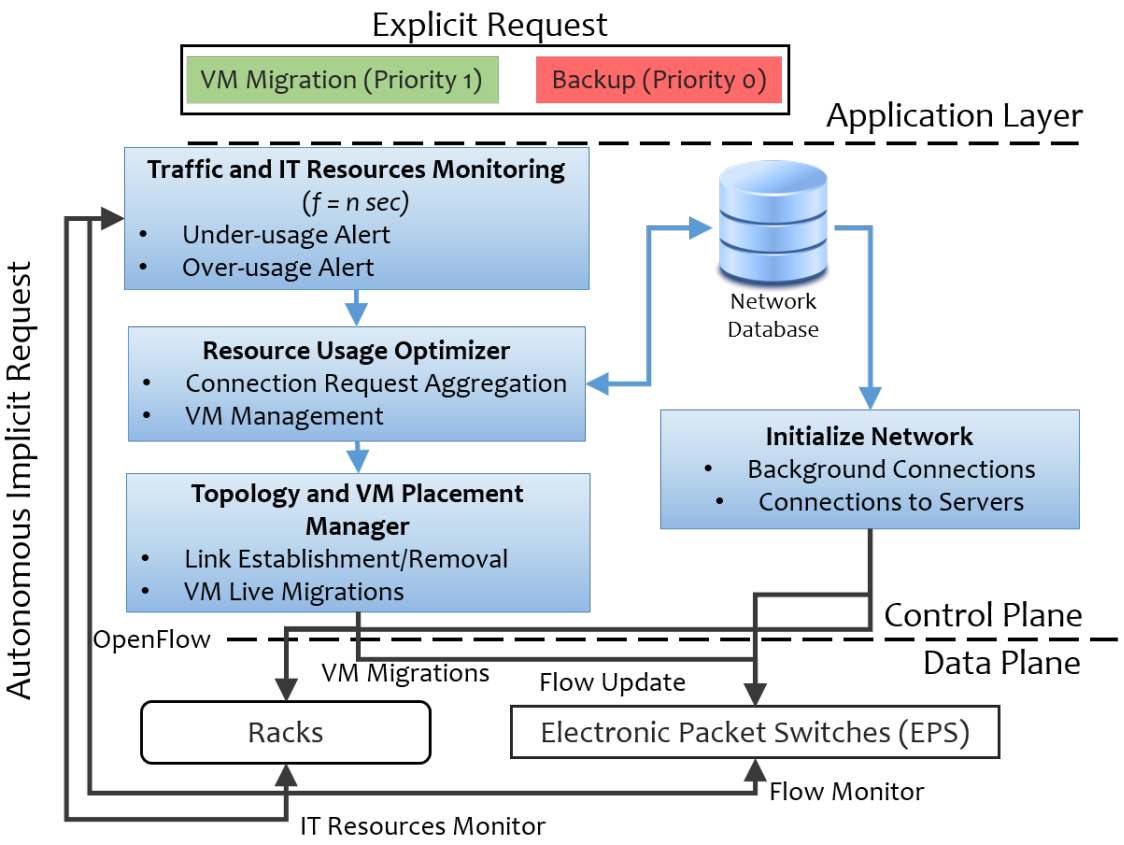


Figure 6.2. SDN control plane workflow showing the system's ability to monitor and provision explicit and autonomous (implicit) network and computational resources

are needed, and whether currently operating VMs need to be consolidated. If yes, the resource usage optimizer invokes the topology and VM placement manager to modify the flow tables of the relevant ToR switches to establish new links, and/or initiates VM live migrations to increase or reduce CPU usage on certain servers. Explicit dynamic connections are made from the application layer directly to the resource usage optimizer module. The details of the decision making process for establishing dynamic links as well as VM migrations are as follows.

The algorithm for the implicit operation of provisioning bandwidth and IT resources is shown in Algorithm 1. To start, the initialization of the network architecture estab-

lishes the inter-DC background connections necessary for basic connectivity, and the controller establishes connections to the servers. After that, periodic monitoring obtains the throughput of the background connections and consumption of IT resources, such as the CPU, memory, or storage usage. Monitoring of the link throughput is performed by obtaining the current and previous values of the number of bytes sent by each flow on the shared link, and dividing this difference by the monitoring interval time.

The first half of Algorithm 1 describes the bandwidth allocation algorithm for the metro network. The SDN controller uses the throughput values calculated through periodic traffic monitoring of each flow to determine whether the traffic on background connections B between source and destination data centers (DC S and DC D) is higher than a specified threshold. If so, the controller will calculate the standard deviation (SD) of the throughput between the different flows in B. If the SD $\leq$ 1, a new background connection is established between DC S and DC D. If the SD > 1, then a new dynamic connection is created for the flow that generates the highest traffic (for optical gateways, a new dynamic connection is created only when wavelength channels are available - more below). This allows a dynamic connection to be allocated only if the largest flow has a significantly higher throughput than the others. Otherwise, if the flows are sharing the bandwidth of the link with an almost equal demand, then allocating a new background connection is fairer to each flow.

Note that for optical gateways, when the controller creates a new dynamic connection, it uses the highest available data rate in the network, limited by the distance between DC S and DC D to be within the optical reach of the transceivers. It also assigns priorities to each new dynamic connection so that higher priority connections can could push

existing lower priority connections. A new dynamic connection can be assigned priority 0 representing a connection for bulk traffic that are tolerant of delays, or priority 1 representing a critical connection that sends delay-sensitive traffic. In such a case, the controller a force an active dynamic connection with priority 0 to move to a lower data rate or to a background connection.

The algorithm for explicit bandwidth allocation (by an operator) is designed for single-rate converged networks and aims at minimizing the blocking probability for dynamic connections with high priority. According to this algorithm, when the metro network controller receives a new explicit dynamic connection request (R) between source data center DC S and destination data center DC D, it runs a routing and wavelength assignment (RWA) algorithm. If the RWA identifies an available lightpath between DC S and DC D, then R is served using that lightpath. If the RWA does not find an available lightpath and R has priority 0, then R is rescheduled at a later time. If the RWA does not find an available lightpath and R has priority > 0, the algorithm tries to move an active connection (R') with lower priority to a background connection. In this way the optical resources occupied by R' can be released and utilized to serve R. Note that this causes an increment in the transfer time for R' due to the fact that using a background connection it will share the channel capacity with other traffic.

The second half of Algorithm 1 describes the mechanism for autonomous VM consolidation. The concept of this algorithm is analogous to the implicit algorithm for bandwidth allocation discussed previously. In the beginning, requests to the DC are handled by VMs consolidated onto a small number of active servers. The SDN controller obtains the current IT resource usage of each server in the distributed DC network at regular intervals.

When the workload in a server increases past a certain threshold causing performance to degrade if left unchecked, the controller will initiate VM live migrations to the next available server to distribute the workload, which can be located within the same rack in the same metro DC, or a rack located in another metro DC, which will also prompt a dynamic link allocation if necessary. This allows the disruption of service caused by lack of network bandwidth for the VM migration to be reduced to a minimum. On the other hand, VM migrations can also be used to reduce traffic hotspots. If a dedicated dynamic link is insufficient to provide the bandwidth necessary to meet demands, the controller will perform VM migrations to a second server to relieve network congestion by allowing part of the traffic to be directed to the second server despite sufficient CPU and memory resources in the first server.

The last ”else if” statement in Algorithm 1 describes the consolidation process, where if the second server V that the VM was migrated to is still active but the original server U where the VM came from is consuming IT resources or dynamic link bandwidth below a certain threshold, meaning that it is no longer consuming as much resources as before, then this VM is migrated back to U to maintain high CPU usage of the original server U. This process allows the second server V to be turned off to save operational costs and energy consumption.

Overall, our control plane ensures that both network resources and IT resources are available to the servers in the distributed data center network at all times to mitigate congestion and maintain high CPU usage, while minimizing the number of active computation nodes. In this way, our architecture eliminates the costs associated with hardware maintenance and energy usage from keeping unnecessary servers active.

Chapter 7

Simulation and Testbed Results

An event-driven simulator was developed to evaluate the benefits of the proposed converged architecture and the dynamic network bandwidth allocation portion of the control algorithm. A testbed was also built to evaluate both the network bandwidth allocation and IT resource management capabilities of the system.

7.1 Simulations and Numerical Results

For our network simulation the topology is composed of 38 nodes, 59 links and 100 wavelengths per fiber. Each node represents a metro data center with 100 racks/pods and each rack/pod switch is equipped with one 10G and one 40G WDM tunable transceivers connected to the optical gateway. The EPS is equipped with 25 10G grey transceivers connected to the optical gateway as well. The control plane employs k-shortest paths with first-fit wavelength assignment (k-SP FF) algorithm to establish new lightpaths. We assume that rack/pod switches generate traffic flows with lognormal inter-arrival distribution. We vary the mean of the lognormal distribution to mimic different traffic loads. In average half of the traffic flows have priority 0 (Bulk) and half have priority 1 (Critical). The flows represent data transfers with sizes uniformly distributed between 1 and 500 GB. We compared the performance of: (i) Converged Multi-Rate (MR) network, (ii) Con-

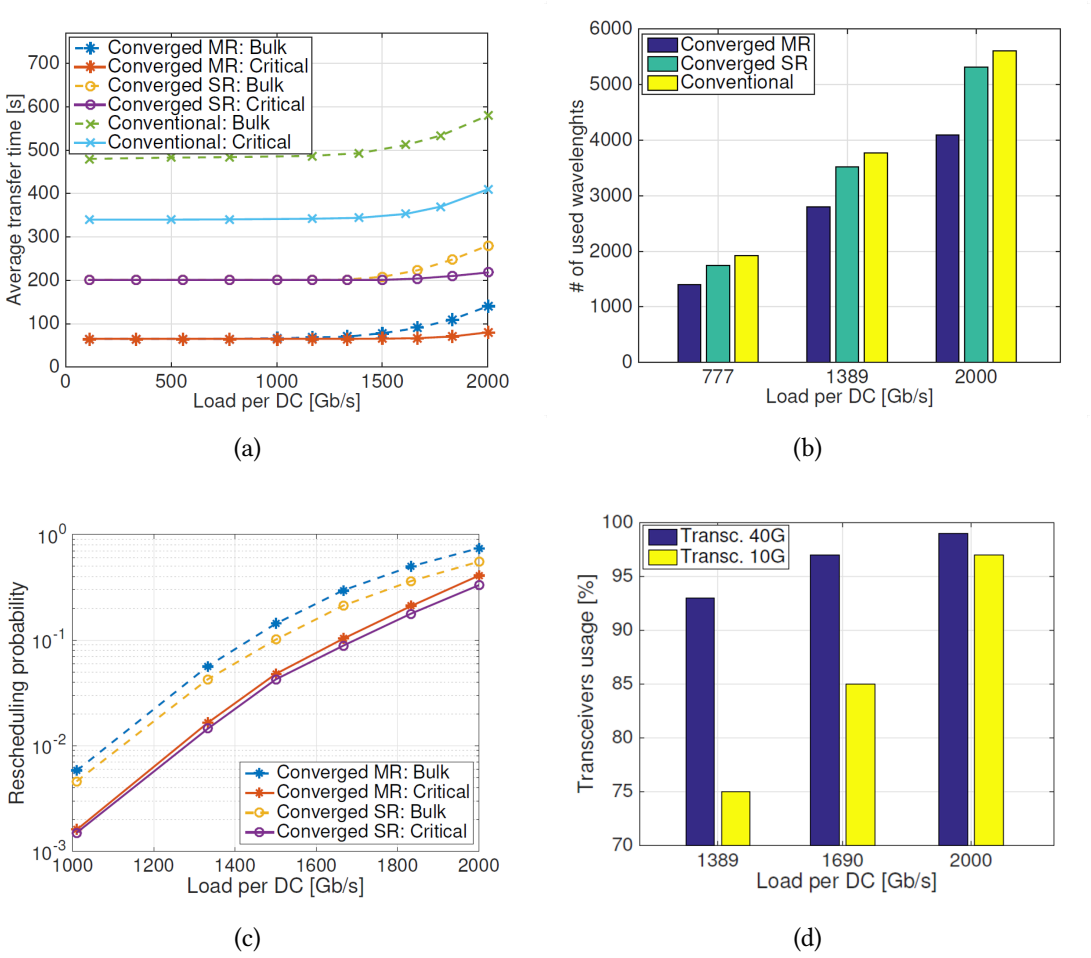


Figure 7.1. Simulation results comparing the performance of the proposed Converged Multi-Rate (MR) and Single-Rate (SR) architectures with implicit control strategy. (a) Average transfer times, (b) average network resource usage, (c) blocking probability for new connections, (d) average usage of transceivers at 10G and 40G

verged Single Rate (SR) and (iii) Conventional SR network. In our simulations the three networks have the same overall capacity. In the Converged MR, 40G transceivers are used for dynamic connections with 10G transceivers are used for background connections. In SR, only 10G transceivers are available and they are used for both background and dynamic connections. The Conventional network relies only on background connections. To manage different levels of priority in the Conventional network, we employ a traffic engineering technique that reserves at least 3 Gbps capacity over a background connec-

tion for Critical transfers. The results are illustrated in Figure 7.1. Figure 7.1(a) shows the average time required to complete a data transfer between different metro data centers, as a function of the load. It can be observed that the proposed Converged MR provides in average 2.5× faster Critical transfers and 2× faster Bulk transfers, with respect to the Converged SR. This is due to the use of multi-rate transmission that allows for improving the transmission efficiency on the dynamic connections. In addition, the Converged MR provides 5× faster Critical and Bulk transfers compared to the Conventional network. This is due to the combined use of multi-rate transmission and dynamic connections for performing large data transfers. On the other hand, the Converged SR network provides in average 2.5× faster Critical transfers and 2× faster Bulk transfers, with respect to the Conventional, thanks to the use of dynamic connections.

Figure 7.1(b) shows the average number of wavelengths required to carry different loads. The Converged MR requires at least 20% less wavelengths than the Converged SR and 25% less wavelengths than the Conventional. On the other hand, the Converged SR requires between 5% and 10% less wavelengths with respect to the Conventional network. Consequently, we can conclude that the Converged networks provide a more efficient resource usage. The main reasons are that in the Converged architectures the data transfers are faster and thus occupy network resources for shorter times. In addition, the dynamic connections are always fully utilized and no bandwidth is wasted.

Figure 7.1(c) shows the blocking probability in the Converged networks. The blocking probability indicates the probability that the establishment of a new connection is blocked due to the lack of wavelength resources or of free transceivers for establishing the lightpath between the source and destination data centers. It can be observed in the

Converged architectures the Critical connections have lower blocking probability than the Bulk connections. This is due to the procedure described in control algorithm that allows to move Bulk connections to background to serve new Critical connections. In the Converged MR the blocking probability is slightly higher than in the Converged SR, due to the fact that less transceivers are employed at the rack/pod switches, which leads to a slightly higher probability that a new connection will not be served due to the lack of a free transceiver.

Finally, in Figure 7.1(d) we show the percentage of the time in which the 10G and 40G transceivers at the racks/pods switches are utilized in the Converged MR network. It is is shown that, for sufficiently high traffic, the 40G transceivers are utilized almost 100% of the time. The reason is that in our control strategy we prioritize the use of the 40G transceivers, which are the most expensive asset, so that the operator can get maximum outcome from the investment.

7.2 Experimental Prototype

A prototype implementation of 3 data centers (2 fully implemented, 1 emulated) was built (Figure 7.2). Data centers 1 and 2 consist of 4 racks, are connected to a server. Servers are equipped with a 6-core Intel Xeon processor, 24 GB memory and a dual-port 10G Network Interface Card (NIC). ToR switches are implemented by logically dividing the two Pica8 10G OpenFlow switches to 8 bridges. Each ToR has a 10G port to the server and a 10G port to the EPS switch with a direct-attached copper cable, and a 10G port to the optical gateway with a 10G optical transceiver. ToR_1 also has a 40G uplink. The EPS is

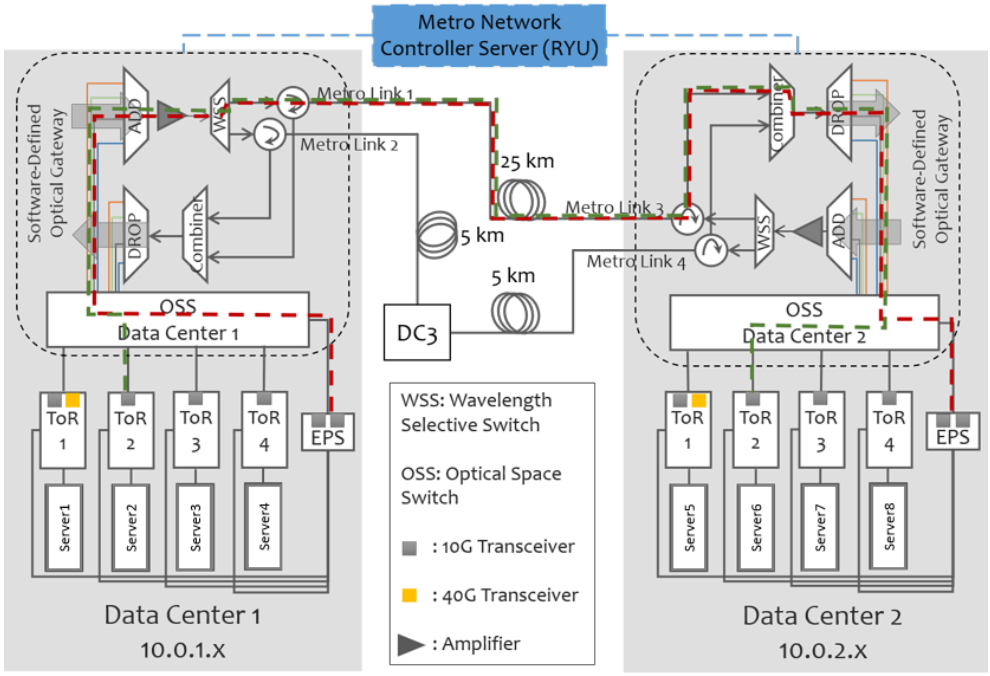


Figure 7.2. Prototype of the 3-node metro-scale converged data center network. DC1 and DC2 are fully implemented, while DC3 is implemented only on the control plane. Distances between the data centers range between 5 to 25km

a bridge in a Pica8 10G OpenFlow switch with two 10G transceivers. Optical gateways are implemented using Calient and Polatis Optical Space Switches (OSS), Nistica Wavelength Selective Switch (WSS) and DWDM Mux/Demux. The 10G optical transceivers are 10G SFP+ DWDM with 24 dB power budget. Due to limitations in form-factor single wavelength DWDM 40G transceivers, we used a (4x10G) QSFP+ with a 18 dB power budget; however commercial single wavelength form-factor 40G, 100G will be available soon [112].

For the control plane, an controller server runs the SDN Ryu OpenFlow controller and is connected to the ToR and optical gateways via 1 Gbps Ethernet campus Internet. Data center 1 and 2 are in 10.0.1.x and 10.0.2.x subnets, respectively. All the control plane modules are developed in Python and are integrated in a RYU application. Data center 3

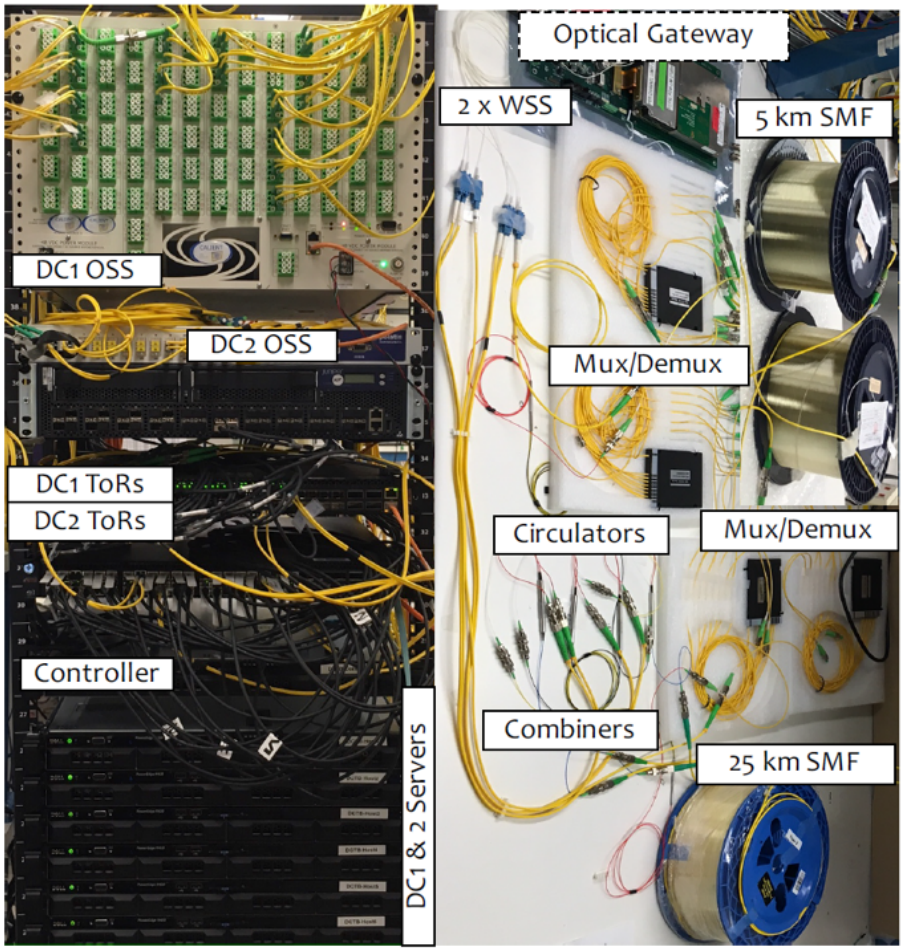


Figure 7.3. Snapshot of the 3-data center metro-network prototype

is only implemented in the control plane to achieve a mesh topology. Distances between the data centers are 5–25 km.

For managing IT resource usage, virtual machines (VMs) are created on each server using the Centos VM Manager with 8 Gb of hard disk and 8 Gb of local memory usage each. IT resource monitoring was performed using the *psutil* Python library. The SDN controller sends messages to the servers over an SSH connection. Each servers contains a script for running the *psutil* function that returns the current system-wide CPU utilization as a percentage, which is then returned to the SDN controller for processing. VM

migrations were performed using the functions from the *libvirt* Python library. Once a decision has been made to migrate a VM, the SDN controller uses the source host and destination host IP addresses to initiate the VM live migration.

7.3 Dynamic Network Bandwidth Allocation

In this section we first demonstrate the testbed prototype's ability to perform dynamic bandwidth allocation only, without the IT resource management capabilities. Figure 7.4(a) shows the evaluation of performing explicit background connections. We show the throughput as a function of time for the background connection between data centers 1 and 2. At the start, only Rack 1 is transmitting data, and thus they use the full 10 Gbps bandwidth of the link. At 10s, Racks 2–4 also start data transmission, so the 10 Gbps link capacity is shared between the 4 Racks. Between, 30-40s, Rack 1 again utilizes the whole bandwidth. At 40s, the background connection is shared between Racks 1 and 2, each 4.5 Gbps and so on. For low priority services with less stringent bandwidth requirements, a background connection performs well enough. However for higher priority services where unpredictable changes on the throughput is not acceptable, a dedicated dynamic connection is required.

Next, we demonstrate explicit dynamic connections between data centers 1 and 2. Four Rack-to-Rack connections are established and the throughput is measured. Figure 7.4(b) shows the result that is captured simultaneously on the servers. All connections are utilizing the maximum available bandwidth of 10 Gbps. The slight difference between the throughput of Rack 4 connection is due to difference in the brand of the transceiver

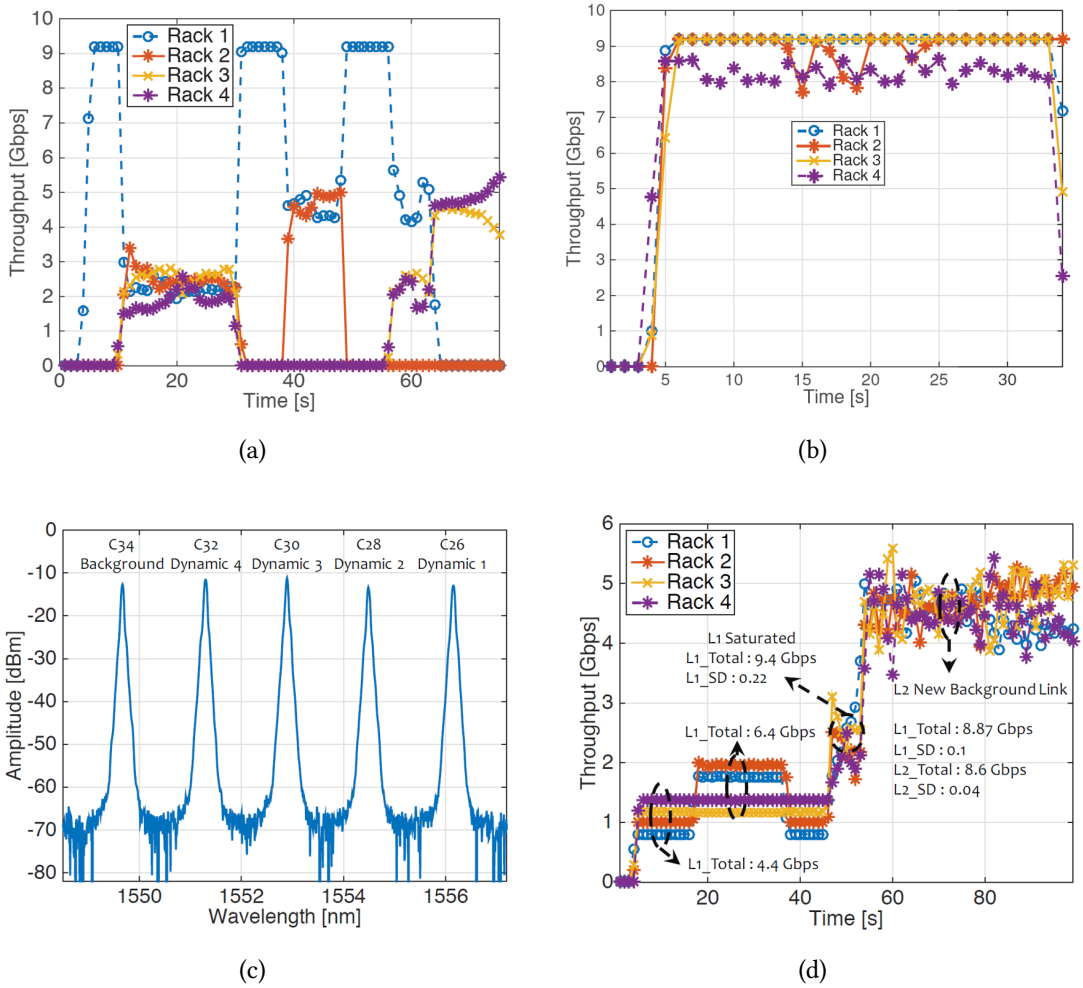


Figure 7.4. Experimental results on the prototype: (a) Random traffic change between 4 racks on a background connection, (b) 4 simultaneous dynamic rack-to-rack connections with an explicit request, (c) Spectrum of DC1 and DC2 links consisting of 5 DWDM connections (1 background and 4 dynamic), (d) Autonomous (Implicit) bandwidth adjustment on a background connection utilized by 4 racks between DC1 and DC2

module compared to the rest. With dynamic connections, strict latency and bandwidth demands for high priority services are guaranteed. Figure 7.4(c) shows the spectrum of the connections after 25km of transmission. Channel C34 is the background connection and channels C26, C28, C30 and C32 are the four dynamic connections.

We continue the protoype evaluation by demonstrating the control plane capability to establish implicit connections. The control plane monitors the 4 flows between Racks

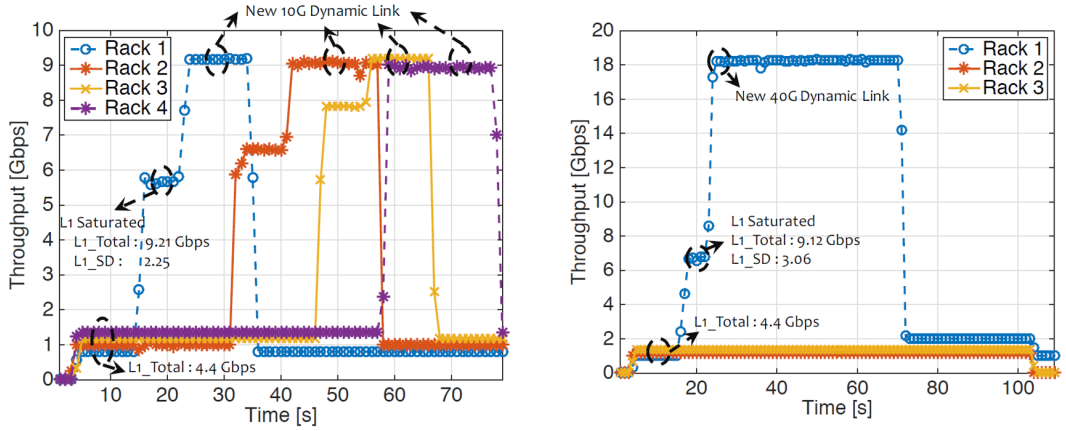


Figure 7.4. (e) Autonomous (Implicit) bandwidth adjustment by establishing a dynamic connection between DC1 and DC2 in a single-rate data plane (10G), (f) Autonomous (Implicit) bandwidth adjustment by establishing a dynamic connection between DC1 and DC2 in a multi-rate data plane (10G and 40G)

1–4 of data centers 1 and 2 on the background connection every 2s and calculates the total throughout of all four. As shown in Figure 7.4(d), at the beginning Racks 1–4 have throughput of 0.8, 1, 1.2 and 1.4 Gbps, respectively, which generates 4.4 Gbps in total. Between 18–36s, the throughput of links 1 and 2 increases; however the overall traffic is still under the threshold that is about 9 Gbps. At 45s, all racks starting transmitting more data that results in link saturation at 50s (total traffic: 9.4 Gbps, SD: 0.22). At this point, the control plane makes a new background connection since the standard deviation (SD) > 1. Now, there are two background connections having total throughput of 8.87 and 8.60 and the SD of 0.1 and 0.04, respectively.

Next, we evaluate the autonomous dynamic connection establishment on a single-rate scenario. Figure 7.5(a) shows the racks' throughput. The experiment starts by having total 4.4 Gbps traffic on the background connection. At 15s, Rack 1 increases its throughput to the point that the background connection is over the threshold of 9 Gbps with the 4

flows having SD of 2.25. Since, it is larger than 1, the control plane establishes a new dynamic connection for the flow with the largest traffic that is Rack 1. At 32s, Rack 2 starts transmitting more data and saturating the background link, thus the control plane established a new dynamic connection for Rack 2 at 42s. Same trend happens for Racks 3 and 4, and they also get a dedicated background connection.

Finally, we demonstrate an implicit dynamic connection in the multi-rate scenario. Racks 1–3 are transmitting data at 0.8, 1.2, and 1.4 Gbps, total 4.4 Gbps on the background connection (Figure 7.5(b)). At 16s, Rack 1 starts increasing the traffic, resulting to background link saturation at 20s. The SD of the 3 flows is > 1, thus the control plane creates a new 40G dynamic link for the flow with the largest traffic that is Rack1. Now the 18 Gbps of traffic from Rack 1 is on the dynamic connection and the traffic from Racks 2 and 3 are on the background connection between the two data centers. At 70s, the traffic from Rack 1 drops to 1 Gbps, hence the dynamic connection is removed and all 3 Racks transmit data on the background connection.

7.4 Combined IT Resource and Dynamic Bandwidth Allocation

This section shows the prototype control plane being aware of not only the network bandwidth between data centers as well as keeping track of the CPU usage of servers as well. We performed two sets of experiments (Figure 7.5) with CPU usage as the limiting IT resource. The top two graphs show the CPU usage percentage over time of the relevant

servers. The bottom two graphs show the throughput of the servers. The top and bottom graphs of each experiment take place simultaneously for the same experimental procedure.

In the first scenario (Figure 7.5(a)), we focus on the behavior of the control plane under varying computational loads. New requests arrive at DC1, causing new VMs to be spun up in Server 1, reflected in increasing CPU usage over time. The CPU usage of Server 1 continues to increase over time as more VMs are consolidated onto it, until at the 54 second mark (shown by the first dotted black line) when the CPU usage grows above a maximum threshold of 80%. This increase triggers a VM migration from Server 1 to Server 5 in DC2, resulting in a decrease in CPU usage of Server 1 while CPU usage increases for Server 5 due to the offset workload. We also observe the effect on the network, shown by the short burst of traffic generated by Server 1 (bottom graph). When the CPU usage of Server 1 has decreased to a sufficient level below 30% while the CPU usage of Server 5 remains at a moderate level of 40%, the controller initiates a migration for this VM to move back to Server 1 to consolidate the workload, (marked by the second dotted black line). We observe that Server 5's CPU usage decreases to 0, while Server 1's CPU usage increases slightly, until the request is completed. The traffic created by the migration back to Server 1 is also shown in the throughput of Server 5 in the bottom graph.

The second experiment (Figure 7.5(b)) shows the effects on CPU usage and network throughput when the consolidation of VMs onto Server 1 leads to high network usage and how the control plane handles this situation. To begin, Server 1 is running VMs that consume a low level amount of CPU capacity and network bandwidth. At the 23 second mark, new requests cause the workload to increase to over 50% and the throughput to

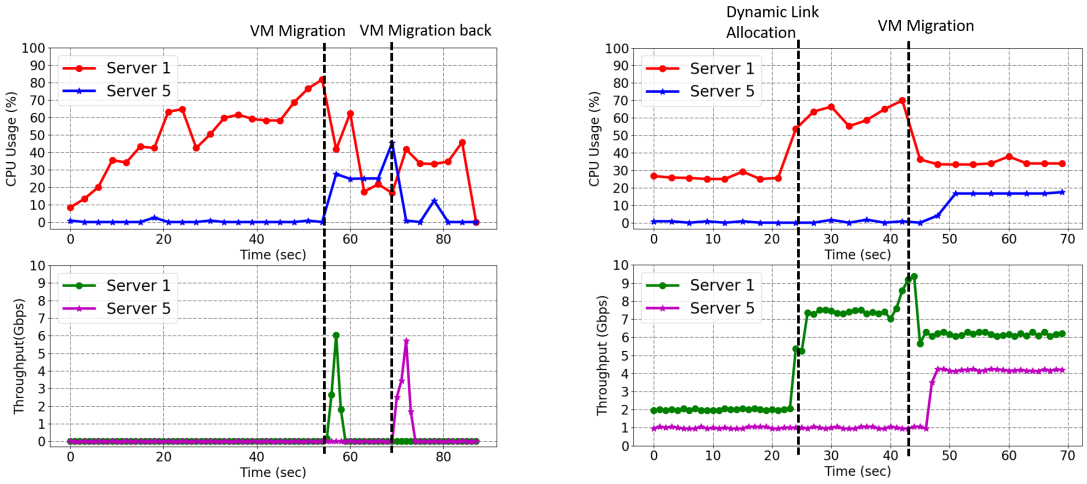


Figure 7.5. Experimental results from the testbed: a) Varying CPU usage triggering implicit VM migrations, and throughput caused by the migrations, b) high bandwidth usage caused by consolidation of VMs causing dynamic link allocations and VM migration to relieve congested links and prevent network hotspots

approximately 7.5 Gbps. The CPU usage is not enough to cause a VM migration, but traffic demands cause a dynamic link to be allocated. As the CPU usage on Server 1 is still low enough for new VMs to be created, the bandwidth usage now grows to near full capacity (90%) of the 10G link, causing a VM migration from Server 1 in DC1 to Server 5 in DC2 at the time marked by the second dotted line at 42 seconds. This VM migration leads to a reduced bandwidth utilization of approximately 6 Gbps by Server 1, and approximately 4 Gbps by Server 5. Server 1 continues to use the dynamic link, while Server 5 uses the background connection, both of which are uncongested. We note that the throughput of Server 1 remains high for approximately 1 second beyond the dotted line instead of dropping immediately due to the additional traffic generated by the VM migration at that time. Additionally, since the VM has migrated to a different physical server, there is a small delay in Server 5's throughput due to the time required to reestablish connection to

the VM.

Overall, we demonstrate the feasibility of our control plane algorithm by subjecting our control plane to handle various different scenarios involving dynamic load variations on CPU usage as well as both background and dynamic link bandwidth usage. In each scenario, the results show that the control plane properly detects high resource utilization for both network bandwidth and CPU usage and performs the appropriate actions to allocate resources when the thresholds are reached. Therefore we show that it is possible to build a reconfigurable and autonomous resource provisioning control plane using commodity electronic switches that can steer bandwidth and consolidate VMs for efficient resource utilization under different application load conditions.

Chapter 8

Conclusion and Discussion

In this part we propose a network architecture designed to manage IT and network resources for metro-scale networks with distributed small to mid-sized data centers. Leveraging SDN, we take advantage of its ability to monitor the entire network domain in order to centrally aggregate statistics and provision both network and IT resources in an approximate and globally optimal way. With dynamic links available to be used on-demand, we provide a simple and effective method to utilize the spare network capacity of the DC to offload traffic from congested links. Along with capability for dynamic bandwidth provisioning, we also developed the capability of IT resource management through VM consolidation for cloud computing applications, using the same principles of monitoring and provisioning on-demand as used for the network. This is described in Chapter 6. From there we developed a large scale simulation of 38 data centers to evaluate the performance of dynamic network provisioning strategy (Chapter 7, which showed 2.5× faster transfer of critical data and 2× faster transfer of bulk data compared to conventional systems, thanks to the use of dynamic connections. We also show numerous other improvements in wavelength channel usage, blocking probability, and link utilization. We also developed an experimental prototype of 3 data centers (2 fully implemented and 1 emulated) with both network and IT resource provisioning capabilities to demonstrate the feasibility

of our concept. We showed that this system autonomously maintains an adequate level of both IT and network resources at all times, and the provisioning of these two types of resources can occur synergistically to realize overall performance improvements and reliability.

8.1 Future Work

While SDN provides reliable monitoring of infrastructure information and efficient provisioning of resources, it also generates extra overhead in the network from gathering the usage statistics of both the switching components and the compute elements in the data centers. This is a well-known challenge for SDN networks due to the centralized nature of the network architecture. A possible approach to address the issue can be the following: the network can be divided among multiple SDN controllers, where each controller will perform the same network bandwidth and IT resource allocation algorithm for the section of the network that it controls. A higher level controller or orchestrator that acts as a mediator for any coordination would be required between the local controllers.

Part III

Lightbridge

Chapter 9

Addressing Network Communication over Optically Switched Networks

9.1 Introduction

In the previous two parts presented in this thesis, both feature network architectures interconnected with optical switches. However, optically switched networks present a new challenge that is previously not faced with electronic interconnects - a switching event in an optical network changes the *physical* topology of the network, due to the its circuit switching nature. This property of optical switches, when incorporated in traditional packet switched environments, leads to a phenomenon where after the photonic switch has performed its switching operation and the lightwave is now transmitting to a different port, the packet switch needs to recognize this new change and reestablish the link. This is equivalent of manually unplugging an optical transceiver and plugging it into a different port. For current commercial packet switches which are designed under the assumption that such an action would never be performed after the initial setup and never while an application is running, it results in a massive delay when an optical switch is introduced. The mechanism in which an EPS recognizes that a new transceiver has been plugged in is located in the operating system of the packet switch, which is constantly polling the state

of its ports every few hundred milliseconds, which corresponds with our own measurements conducted in Part 1 of this thesis (Figure 4.1 and Figure 4.3(b)), which constitutes an overwhelming part of the overall end-to-end switching latency. In addition, optical switches lack the fundamental ability to buffer packets unlike electronic packet switches. While the topic of optical buffering is a heavily researched topic, approaches are generally limited by component complexity or in the case of slow light devices, the inability to reach the delay time and data rates required of a practical optical buffer [113].

From these fundamental characteristics, a crucial problem arises in whether it is safe or not to send a packet at any given point in time in an optically switched network by any end host. The answer to this question critically depends on coordinated switching, routing and forwarding operations across the network.

In this part of the thesis we introduce Lightbridge, a software platform designed for optically switched networks to arbitrate circuit contention safely and efficiently in real-time. It operates in the problem space where there are more optical circuits that are desired than can be physically realized at any given point in time, and provides the state of the interconnect network topology to the processing nodes, so that transmission is only performed when the physically link is available.

Specifically, Lightbridge is composed of two major parts:

1. A Linux kernel module that integrates the nuances of optically circuit switched networks into the networking subsystem.
2. A distributed control plane for realizing safe and efficient communications over optically switched networks.

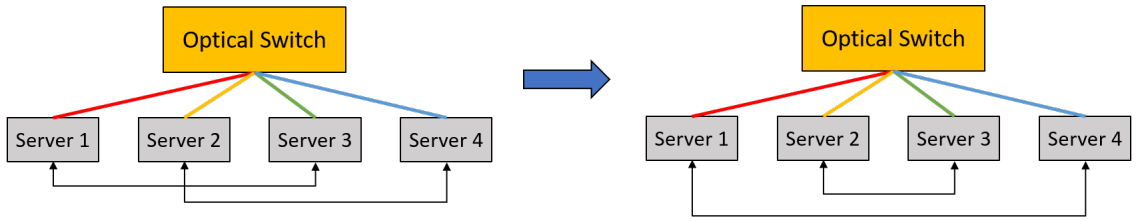


Figure 9.1. Change in optical switch state requires arbitration by the control plane to remove previous connections and buffering of packets whose link is unavailable

The Lightbridge design decomposes the problem space into local buffer management for each optical circuit, and global circuit coordination. Buffer management is handled by the Lightbridge kernel module while the coordination of the global circuits is managed by the distributed control plane which will have the capability to provide network information to each host node. The general premise is that when a host wants to send packets over the network, it needs to first check whether its outgoing link is available or not, which is dictated by the state of the optical switch that the circuit is connected to. If the circuit is not available, then any packets sent over this circuit will be dropped due to the optical loss. Therefore, these packets must be buffered until the state of the optical switch has changed to a state where the given circuit for this host has become available, upon which the packets stored in the buffer can be transmitted. Additionally, when more circuits are required than are physically possible at a given time, scheduling must take place. Hosts can also request circuits to the optical switch controller which arbitrates requests through a consistent scheduling protocol.

A simple example is shown in Figure 9.1. Four end-host servers 1-4 are connected to an optical switch that is capable of creating only two circuits at a time between the four connections. Each colored line represents a physical communication line, which can be

an individual fiber strand (in the case of spatial switching) or a lambda for wavelength switching. The initial state of this network connects Servers 1 and 3, and Servers 2 and 4. Now suppose that Server 1 needs to communicate with Server 4. This requires a switching event to take place, which will remove the current circuit from Server 1 to 3. Therefore, any packets that needs to transmit between Servers 1 and 3 needs to be buffered.

During a switching event, all end hosts must first confirm that they are ready for the switch controllers to cause the switch to change states beforehand. At the end hosts this asserts that they are now buffering packets for the circuit that will become unavailable once the switch occurs. If there are multiple switch controllers then there will need to be coordination between them as well.

9.2 Lightbridge Components

Overview

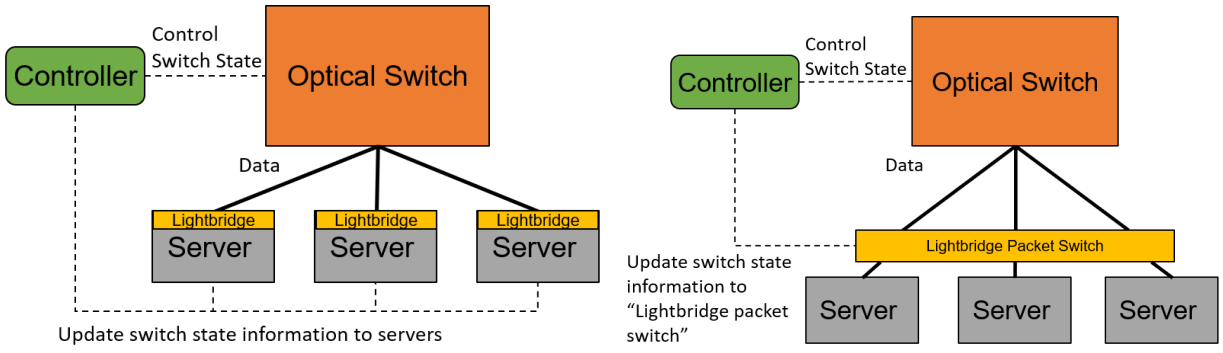


Figure 9.2. How Lightbridge fits into an optically switched network: a) Lightbridge located on each individual end host, and information about the network state is conveyed to each Lightbridge module by the switch controller b) Lightbridge functionality aggregated into a single intelligent packet switch, so that the packet switch Lightbridge functionality on each of its ports

Figure 9.2 shows the general components of the Lightbridge system. Conventional servers are able to communicate with each other through an optical switch, but with the Lightbridge modules loaded within the kernel to provide additional functionalities. The switch controller manage the state of the optical switch as well as convey the switch state information to the servers. There are two different versions of this setup. For Figure 9.2(a), the Lightbridge module is located on each individual server, while in Figure 9.2(b), the LB modules of each server have been aggregated into one unit. Instead of LB modules on each server network interface card (NIC), it will be on a packet switch and each port of the packet switch will be aware whether its optical I/O is currently available or not. This architecture allows for the controller to connect to a single device that aggregates all the traffic instead of having to communicate with each individual host, and subsequently realizes an ”intelligent” packet switch that is aware of the optical network topology. Meanwhile, the servers are off-the-shelf general purpose machines that can be installed in a plug-and-play manner without any modifications. As the Lightbridge module is a Linux-based kernel submodule, it is simple to implement in current packet switches, which use Linux operating systems.

Linux Kernel Submodules

In this subsection the various subcomponents comprising the Lightbridge Linux module is described in detail. The external switch controller is not described in this thesis, as it has yet to be implemented. Only the Linux module component will be discussed.

The software modules constituting the Lightbridge Linux module functionality is in

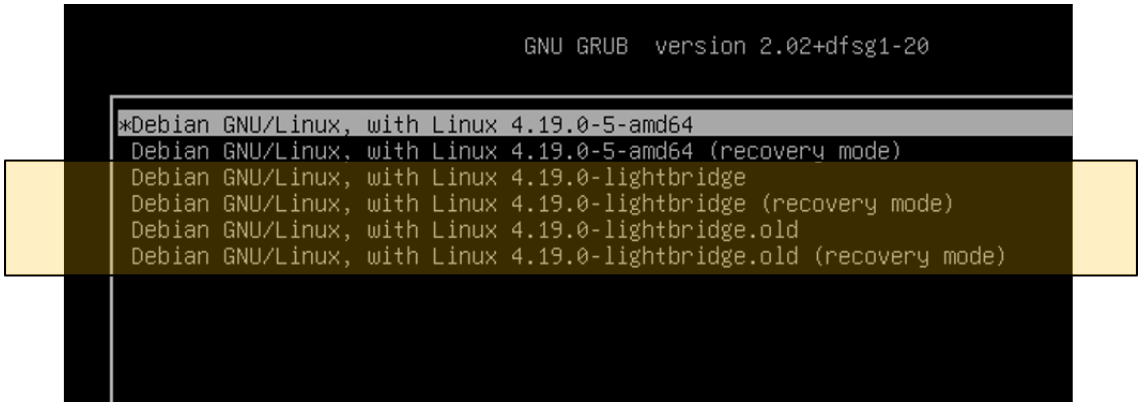


Figure 9.3. Bootup menu showing kernel modified with Lightbridge

two parts.

1. A Loadable Kernel Module (LKM) that is loaded into the kernel on demand, and is responsible for the transmission and buffering functionalities
2. A secondary module that is a direct modification of the network protocol section of the Linux kernel, responsible for handling packet reception to the end host.

The LKM and the secondary module are part of an in-tree build, meaning a modified kernel needs to be selected during bootup sequence (shown in Figure 9.3). It can be seen that below the standard Debian kernel, the highlighted box shows the modified Debian kernel options that has Lightbridge loaded.

When the LKM is loaded, it creates virtual network interfaces that sits on top of the real physical interface. To test the functionality of this operation, we create 2 Lightbridge (LB) devices, LB1 and LB2, which can be seen in Figure 9.4. Each LB device acts to control the transmission and buffering of packets of a single optical channel, by being a virtual network interface that connects to the physical network interface of the optical channel.

Figure 9.5 shows the operation workflow of the main LKM module. Starting at the

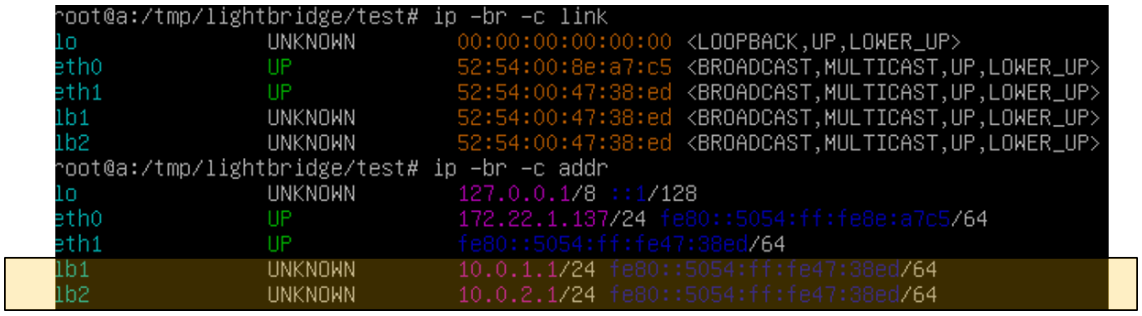


Figure 9.4. Lightbridge devices lb1 and lb2 in server A, which are virtual devices with their own IP addresses

top is the initialization of LB devices, a lookup table, buffers, as well as establishing the connection to the parent interface. The lookup table is the means in which the end hosts are able to know the availability of its optical circuit, and is also what is being constantly updated by the switch controller. The implementation of the lookup table is simply a single 64 bit integer, serving as a bitmap to indicate which links are available or not. From there, two situations can arise. First, if a packet needs to be transmitted, the lookup table is checked to see if the circuit for the transmitting server is available. If it is, then the packet is transmitted. If it is not, the packet is buffered in a circular buffer (*circ_buf*) data structure until it is filled, upon which the packets are dropped.

The other situation is when the lookup table is modified by the controller. In this case, after the lookup table has been updated, we check to see if the buffers have data queued. If there are buffers that have data queued and the optical circuit corresponding to this buffer has become available, then we can transmit packets from those buffers until they are empty or until the circuit becomes unavailable again. This process repeats until the module exits, upon which the LB devices and buffers are freed.

The workflow for the secondary module responsible for packet reception is shown in

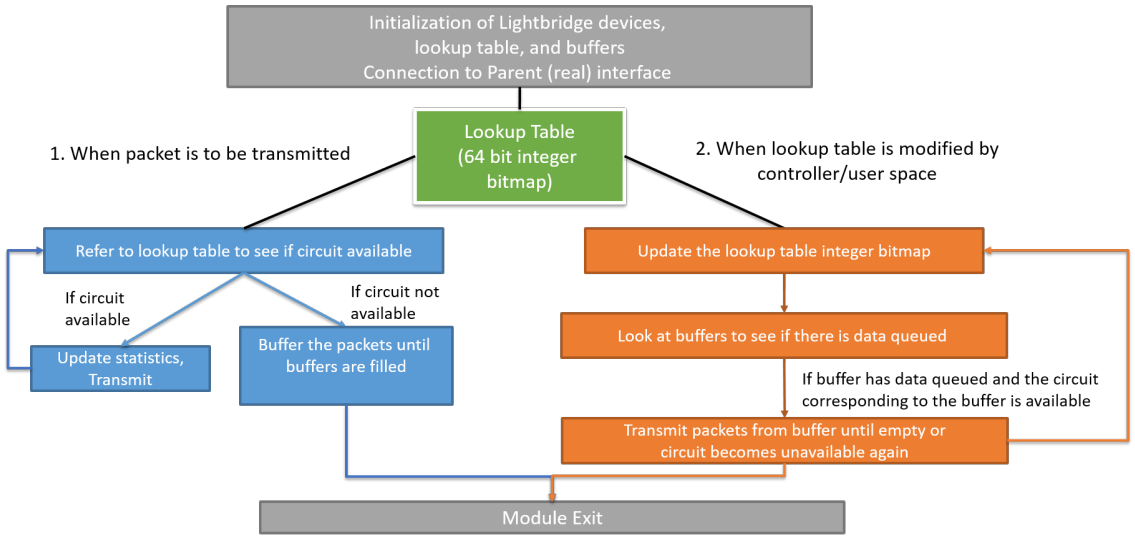


Figure 9.5. Kernel module (LKM) workflow for packet transmission and buffering

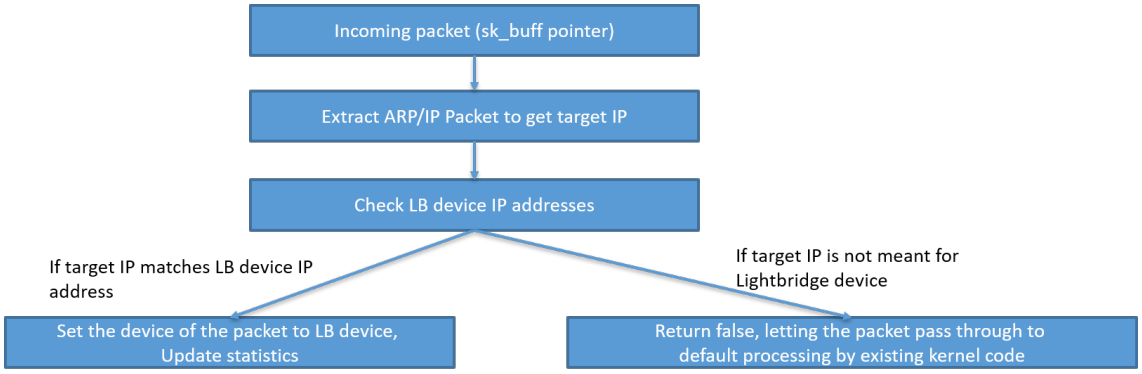


Figure 9.6. Workflow for packet reception and processing by the Lightbridge device

Figure 9.6. It is called upon in the default file in the Linux Debian kernel (*net/core/dev.c*) which handles packet reception (header processing), and is inserted at a point before the default packet reception code so that it intercept the incoming packet so that it can be processed to see if the packet is meant for a Lightbridge device. The process begins with an incoming packet, upon which we perform header processing to extract its target IP and whether it is an ARP or IP packet. We match the target IP with the LB device IP addresses to see if the packet is meant for an LB device. If it is, then we set the device of the packet

to the LB device, and update system statistics. If not, we return false and let the packet pass through the default processing procedure by the existing kernel code.

9.3 Functionality Verification

The functionalities of the two modules were tested in a virtual environment with two virtual machines that communicate with each other through a virtual optical link. A user side module was also written to act as a emulated controller that can update the lookup table bitmap on the kernel side. The following operations were performed to prove the successful implementation of the aforementioned functionalities.

1. Server A and Server B are present, with Server A having LB1 and LB2 devices activated by loading the LKM using the *modprobe* command. LB1 and LB2 have the IP addresses 10.0.1.1 and 10.0.2.1, respectively, while Server B has interfaces with IP addresses 10.0.1.2 and 10.0.2.2, respectively.
2. We first verify that both LB devices can ping their corresponding IP addresses on Server B in the same subnet.
3. We then use the user-side program to set the lookup table so that the circuit for LB1 is blocked, and verify that pinging 10.0.1.2 no longer works.
4. When we use the user program to change the state of the lookup table so that LB1 is no longer blocked and LB2 becomes blocked, we use packet tracing software *tcpdump* to see that all the packets buffered at LB1 is suddenly sent all at once to 10.0.1.2, while packets to 10.0.2.2 from LB2 is blocked.

5. We further verify that the transmission blocking and buffering capabilities, as well as releasing the buffered data when the lookup table is updated works properly by using *netcat* to create a socket to send short text messages from Server A to Server B. We verify that when the circuit is not available, Server B does indeed not receive the messages, and when the circuit becomes available, all the text messages are suddenly received at Server B. This shows that the correct payload is successfully delivered. A screenshot of this test is shown in Figure 9.7.

In Figure 9.7, the top left screen shows the kernel status messages from the *dmesg* command, updating us on the current status of the lookup table. The top right screen is the user side program that is used to manually update the lookup table in Server A, so that it is emulating a switch controller. The bottom left screen shows Server A pinging and sending text messages through *netcat*, and finally, the bottom right screen shows the monitoring of received packets on Server B side using *tcpdump* as well as the receiving side of the *netcat* socket.

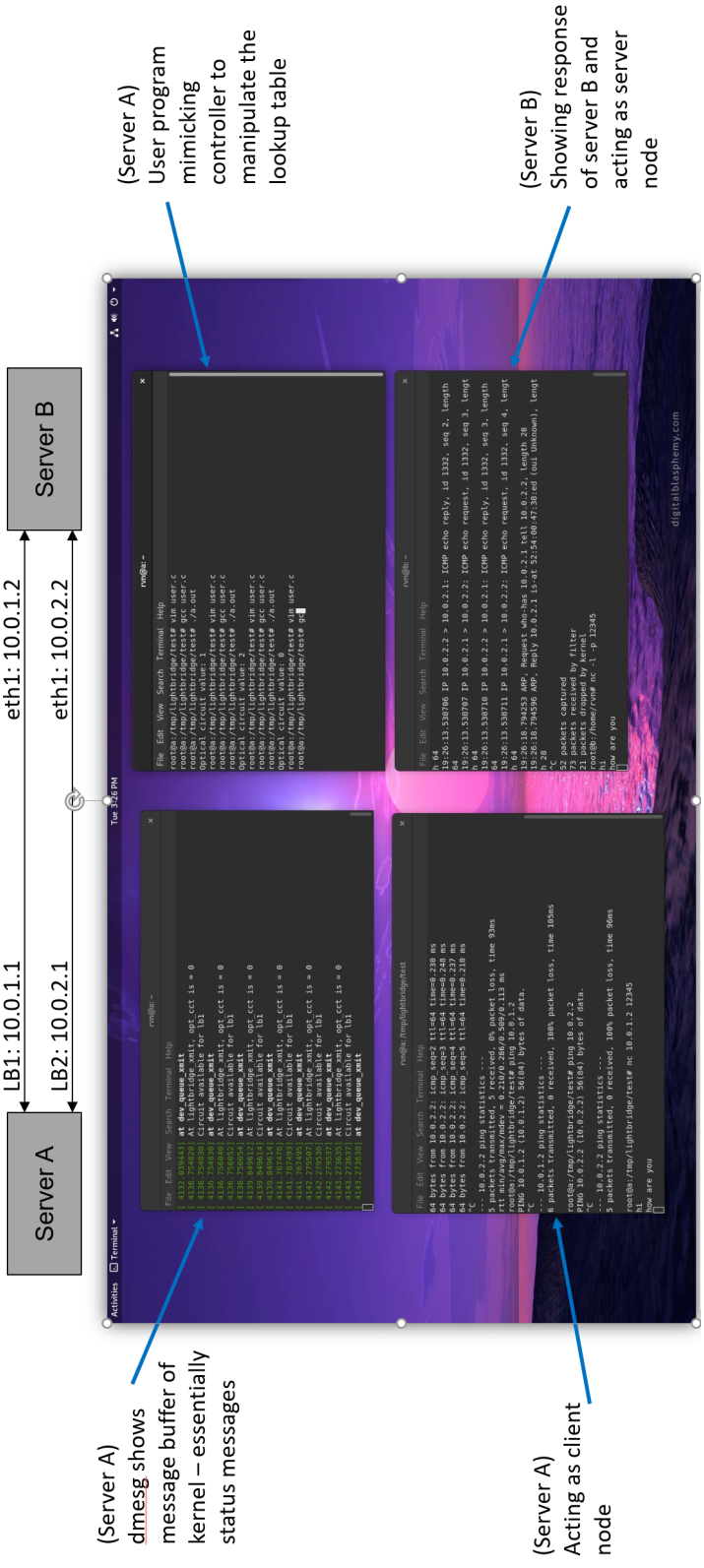


Figure 9.7. Screenshot of Lightbridge tests conducted to verify its functionalities.

Chapter 10

Conclusion and Discussion

This final part of the thesis presented the development of a software solution to address problem of coordination and control of transmitting packets over an optically switched network. As an optical switch changes the physical topology of the network without having the capability to buffer packets, recklessly sending packets by hosts can result in large quantities of packets being dropped and having to be resent, among other issues. We developed Lightbridge, which consists of an external control plane that provides information about the state of the optical switch to the end hosts, as well as kernel-side modifications to the hosts that utilize this information to halt transmission and buffer outgoing packets when the optical circuit is unavailable. It also controls packet reception from the physical interfaces that are connected to the optical switch. This allows packets incoming packets to be properly received, and outgoing packets to be sent only when the optical network is in the proper state.

According to the experiments described and depicted in Figure 9.7, we verify that the Linux Debian kernel has been appropriately modified so that all transmitting packets will first check whether the optical circuit is available, and buffering of the packet occurs when the circuit is not available. The blocking and buffering functionality of the virtual LB devices have also been verified. On the packet reception end we see that incoming

packets that are meant for the LB devices are appropriately handled, while packets on other interfaces pass through unaffected.

10.1 Future Work

While currently we have only implemented the basic functionalities of Lightbridge on the end hosts, future plans will involve the implementation of the distributed control plane that will not only control the state of the optical switch but also convey the state of the optical switch to the end hosts.

Final Conclusion and Remarks

The work presented in this dissertation described various reconfigurable network architectures that leveraged optical switches to provide increased functionality, flexibility and performance improvements. Optical switches have the fundamental capability to direct high bandwidth and low energy optical signals and are crucial for next generation network interconnects, whether it is on the intra-data center scale to the metro and long haul scale. Their integration with current compute systems such as high-performance computing and data centers require a system level control plane that can manage the various hardware of the system to work synergistically with the optical switch, and various other software and hardware modules to integrate the optical switch within the traditional Datacom environment of servers and electronic packet switches. After developing these integration methodologies and demonstrating the capability to relieve congestion in links with bandwidth steering techniques in Part 1, we made the observation that there was a mismatch between the application traffic pattern and the network topology due to the skewness of many high-performance computing applications, which leads to suboptimal performance of the compute system, as the processing nodes were either constantly starved of data to process due to congested links, as well as wasting energy on links that were on but were severely underutilized. With these observations, we determined

how to utilize energy efficient silicon photonic switches within the network infrastructure in order to construct flexible network topologies that can adapt to the traffic demands of the application. This lead to optimized bandwidth allocation and reduced latencies in both Dragonfly and Fat-Tree topologies, which are commonly used in both high-performance computing systems and data centers. By constructing various testbeds that were miniature supercomputers and operating an open-source HPC benchmark application called GTC on the testbed, we were able to show significant performance improvement by observing the total application execution time.

Optical switches are also a crucial component of metro-scale geographically distributed data centers. Such data centers have been deployed rapidly in the last decade due to the explosive increase in traffic demand due to the emergence of novel applications such as the 5G mobile network, data analytics and cloud computing. In order for these data centers to meet the required quality of service demands, it is necessary for an intelligent control plane to manage the allocation of bandwidth and compute resources. In Part 2 we developed such a control plane and testbed to demonstrate the feasibility of a converged intra- and inter-data center network architecture that provided dynamic bandwidth allocation using optical links based on active monitoring of bandwidth and compute resources. This system demonstrated the capability to provide links to support various quality of service demands in bandwidth and latency, but also worked in tandem with autonomous IT resource management through the migration and consolidation of virtual machines.

Finally, in Part 3 we addressed one of the fundamental issues of optical switching, which arises from the fact that optical switches change the physical topology of the net-

work and lack buffering capabilities, meaning that when an optical switch performs a switching operation, packets in transit will be dropped and current systems relies on the TCP protocol of retransmission to allow for communication to continue after the switch operation has been performed. Current packet switches also take a long time to re-establish a link once it goes down and is reconnected, caused by the optical switching operation. Therefore, we developed Lightbridge, which is a modification to the Linux kernel that creates virtual network devices to control the transmission and reception of packets on an optical circuit. The virtual network devices will also be connected to a central controller that will update the end hosts on the state of the network so that hosts will buffer packets instead of transmitting blindly when the optical circuit is not available. This provides a more seamless integration between optically switched networks and traditional equipment including servers and electronic packet switches.

All in all, this thesis provides a detailed exploration on concepts, software and hardware components, and evaluations on the topic of integrating optical circuits within packet switched environments in order to realize next generation compute systems.

Bibliography

[1] P. M. Kogge and J. Shalf, "Exascale computing trends: Adjusting to the "new normal" for computer architecture," *Computing in Science and Engineering*, vol. 15, pp. 16–26, 2013.

[2] *Top500*, https://www.top500.org, (Accessed on 09/10/2018).

[3] S. Rumley, M. Bahadori, R. Polster, S. D. Hammond, D. M. Calhoun, K. Wen, A. Rodrigues, and K. Bergman, "Optical interconnects for extreme scale computing systems," *Parallel Comput.*, vol. 64, no. C, pp. 65–80, May 2017, ISSN: 0167-8191. DOI: 10.1016/j.parco.2017.02.001. [Online]. Available: https://doi.org/10.1016/j.parco.2017.02.001.

[4] *System overview – oak ridge leadership computing facility*, https://www.olcf.ornl.gov/for-users/system-user-guides/summit/system-overview/, (Accessed on 10/25/2019).

[5] J. J. Dongarra, M. A. Heroux, and P. Luszczek, "Hpcg benchmark a new metric for ranking high performance computing systems," 2015.

[6] *Https://www.hpcg-benchmark.org/custom/index.html?lid=158&slid=281*, https://www.hpcg-benchmark.org/custom/index.html?lid=158&slid=281, (Accessed on 12/15/2019).

[7] J. Wang, S. Basu, C. McArdle, and L. P. Barry, "Large-scale hybrid electronic/optical switching networks for datacenters and hpc systems," *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pp. 87–93, 2015.

[8] S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, and J. Shalf, "Reconfigurable hybrid interconnection for static and dynamic scientific applications," in *Proceedings of the 4th International Conference on Computing Frontiers*, ser. CF '07, Ischia, Italy: ACM, 2007, pp. 183–194, ISBN: 978-1-59593-683-7. DOI: 10.1145/1242531.1242559. [Online]. Available: http://doi.acm.org/10.1145/1242531.1242559.

[9] K. Wen, P. Samadi, S. Rumley, C. P. Chen, Y. Shen, M. Bahadori, K. Bergman, and J. Wilke, "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in

SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2016, pp. 166–177. DOI: 10.1109/SC.2016.14.

[10] Y. Shen, X. Meng, Q. Cheng, S. Rumley, N. Abrams, A. Gazman, E. Manzhosov, M. S. Glick, and K. Bergman, "Silicon photonics for extreme scale systems," *J. Lightwave Technol.*, vol. 37, no. 2, pp. 245–259, 2019. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-37-2-245.

[11] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P. Bremer, "Analyzing network health and congestion in dragonfly-based supercomputers," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 93–102. DOI: 10.1109/IPDPS.2016.123.

[12] *Wireline link survey*, https://web.engr.oregonstate.edu/~anandt/linksurvey/, (Accessed on 08/26/2018).

[13] W. Che, Y. F. Tang, J. Zhang, and Y. L. Chow, "Formulas of dielectric and total attenuations of a microstrip line," *Radio Science*, vol. 45, no. 5, 2010. DOI: 10.1029/2009RS004246. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2009RS004246. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2009RS004246.

[14] L. Robert, A. James, B. Keren, B. Shekhar, C. William, C. Laura, C. George, C. Robert, D. William, D. Jack, G. Al, G. Gary, H. Rud, H. Jeffrey, H. Adolfy, K. Dean, K. Peter, L. Richard, S. Vivek, S. Robert, S. John, S. Thomas, and S. Rick, *Top ten exascale research challenges*, DOE ASCAC Subcommitte Report, DOE ASCAC Subcommitte Report, 2014.

[15] C. A. Thraskias, E. N. Lallas, N. Neumann, L. Schares, B. J. Offrein, R. Henker, D. Plettemeier, F. Ellinger, J. Leuthold, and I. Tomkos, "Survey of photonic and plasmonic interconnect technologies for intra-datacenter and high-performance computing communications," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2758–2783, 2018, ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2839672.

[16] M. O'Connor, N. Chatterjee, D. Lee, J. Wilson, A. Agrawal, S. W. Keckler, and W. J. Dally, "Fine-grained dram: Energy-efficient dram for extreme bandwidth systems," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17, Cambridge, Massachusetts: ACM, 2017, pp. 41–54, ISBN: 978-1-4503-4952-9. DOI: 10.1145/3123939.3124545. [Online]. Available: http://doi.acm.org/10.1145/3123939.3124545.

[17] J. Lee, D. Shin, Y. Kim, and H. Yoo, "A 17.5 fj/bit energy-efficient analog sram for mixed-signal processing," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 1010–1013. DOI: 10.1109/ISCAS.2016.7527414.

[18] Y. Arakawa, T. Nakamura, Y. Urino, and T. Fujita, "Silicon photonics for next generation system integration platform," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 72–77, 2013, ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6476868.

[19] Q. Cheng, M. Bahadori, M. Glick, S. Rumley, and K. Bergman, "Recent advances in optical technologies for data centers: A review," *Optica*, vol. 5, no. 11, pp. 1354–1370, 2018. DOI: 10.1364/OPTICA.5.001354. [Online]. Available: http://www.osapublishing.org/optica/abstract.cfm?URI=optica-5-11-1354.

[20] Q. Cheng, M. Bahadori, and K. Bergman, "Advanced path mapping for silicon photonic switch fabrics," in *Conference on Lasers and Electro-Optics*, Optical Society of America, 2017, SW1O.5. DOI: 10.1364/CLEO_SI.2017.SW1O.5.

[21] B. G. Lee, "Photonic switch fabrics in computer communications systems," in *Optical Fiber Communication Conference*, Optical Society of America, 2018, Th3C.3. DOI: 10.1364/OFC.2018.Th3C.3. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2018-Th3C.3.

[22] T. Chu, L. Qiao, W. Tang, D. Guo, and W. Wu, "Fast, high-radix silicon photonic switches," in *Optical Fiber Communication Conference*, Optical Society of America, 2018, Th1J.4. DOI: 10.1364/OFC.2018.Th1J.4. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2018-Th1J.4.

[23] Y. Kawajiri, N. Nemoto, K. Hadama, Y. Ishii, M. Makihara, and J. Yamaguchi, "512 x 512 port 3d mems optical switch module with toroidal concave mirror," *NTT Technical Review*, vol. 10, Nov. 2012.

[24] *Metro network traffic will rise 560% by 2017, study says - dcd*, https://www.datacenterdynamics.com/news/metro-network-traffic-will-rise-560-by-2017-study-says/, (Accessed on 12/15/2019).

[25] *Cisco visual networking index: Forecast and trends, 2017–2022 white paper - cisco*, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html, (Accessed on 10/28/2019).

[26] P. Öhlén, B. Skubic, A. Rostami, M. Fiorani, P. Monti, Z. Ghebretensaé, J. Mårtensson, K. Wang, and L. Wosinska, "Data plane and control architectures for 5g transport networks," *Journal of Lightwave Technology*, vol. 34, no. 6, pp. 1501–1508, 2016, ISSN: 1558-2213. DOI: 10.1109/JLT.2016.2524209.

[27] *The future of data centers - cb insights research*, https://www.cbinsights.com/research/future-of-data-centers/, (Accessed on 10/28/2019).

[28] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "Ease: Epc as a service to ease mobile core network deployment over cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, 2015. DOI: 10.1109/MNET.2015.7064907.

[29] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," *SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 77–88, Jun. 2008, ISSN: 0163-5964. DOI: 10.1145/1394608.1382129. [Online]. Available: http://doi.acm.org/10.1145/1394608.1382129.

[30] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 220–231, Jun. 2009, ISSN: 0163-5964. DOI: 10.1145/1555815.1555783. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555783.

[31] N. Jain, A. Bhatele, X. Ni, N. J. Wright, and L. V. Kale, "Maximizing throughput on a dragonfly network," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 336–347. DOI: 10.1109/SC.2014.33.

[32] G. Michelogiannakis, N. Jiang, D. Becker, and W. J. Dally, "Channel reservation protocol for over-subscribed channels and destinations," in *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013, pp. 1–12. DOI: 10.1145/2503210.2503213.

[33] N. Jiang, D. U. Becker, G. Michelogiannakis, and W. J. Dally, "Network congestion avoidance through speculative reservation," in *IEEE International Symposium on High-Performance Comp Architecture*, 2012, pp. 1–12. DOI: 10.1109/HPCA.2012.6169047.

[34] M. García, E. Vallejo, R. Beivide, M. Odriozola, and M. Valero, "Efficient routing mechanisms for dragonfly networks," in *2013 42nd International Conference on Parallel Processing*, 2013, pp. 582–592. DOI: 10.1109/ICPP.2013.72.

[35] A. Bhatele, W. D. Gropp, N. Jain, and L. V. Kale, "Avoiding hot-spots on two-level direct networks," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–11. DOI: 10.1145/2063384.2063486.

[36] B. Prisacari, G. Rodriguez, P. Heidelberger, D. Chen, C. Minkenberg, and T. Hoefler, "Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks," in *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, ser. HPDC '14, Vancouver, BC, Canada: ACM, 2014, pp. 129–140, ISBN: 978-1-4503-2749-7. DOI: 10.1145/2600212.2600225. [Online]. Available: http://doi.acm.org/10.1145/2600212.2600225.

[37] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There goes the neighborhood: Performance degradation due to nearby jobs," in *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013, pp. 1–12. DOI: 10.1145/2503210.2503247.

[38] J. Estarán, E. Dutisseuil, H. Mardoyan, G. de Valicourt, A. Dupas, Q. P. Van, D. Verchere, B. Uscumlic, P. Dong, Y. Chen, S. Bigo, and Y. Pointurier, "Cloud-boss intra-data center network: On-demand qos guarantees via ub optical slot switching," in *2017 European Conference on Optical Communication (ECOC)*, 2017, pp. 1–3. DOI: 10.1109/ECOC.2017.8346190.

[39] A. Gazman, C. Browning, M. Bahadori, Z. Zhu, P. Samadi, S. Rumley, V. Vujicic, L. P. Barry, and K. Bergman, "Software-defined control-plane for wavelength selective unicast and multicast of optical data in a silicon photonic platform," *Opt. Express*, vol. 25, no. 1, pp. 232–242, 2017. DOI: 10.1364/OE.25.000232. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-25-1-232.

[40] R. Aguinaldo, A. Forencich, C. DeRose, A. Lentine, D. C. Trotter, Y. Fainman, G. Porter, G. Papen, and S. Mookherjea, "Wideband silicon-photonic thermo-optic switch in a wavelength-division multiplexed ring network," *Opt. Express*, vol. 22, no. 7, pp. 8205–8218, 2014. DOI: 10.1364/OE.22.008205. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-22-7-8205.

[41] T. Shiraishi, Q. Li, Y. Liu, X. Zhu, K. Padmaraju, R. Ding, M. Hochberg, and K. Bergman, "A reconfigurable and redundant optically-connected memory system using a silicon photonic switch," in *Optical Fiber Communication Conference*, Optical Society of America, 2014, Th2A.10. DOI: 10.1364/OFC.2014.Th2A.10. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2014-Th2A.10.

[42] K. Ishii, T. Inoue, and S. Namiki, "Toward exa-scale optical circuit switch interconnect networks for future datacenter/HPC," in *Next-Generation Optical Networks for Data Centers and Short-Reach Links IV*, A. K. Srivastava, Ed., International Society for Optics and Photonics, vol. 10131, SPIE, 2017, pp. 12 –19. DOI: 10.1117/12.2250796. [Online]. Available: https://doi.org/10.1117/12.2250796.

[43] D. Nikolova, S. Rumley, D. Calhoun, Q. Li, R. Hendry, P. Samadi, and K. Bergman, "Scaling silicon photonic switch fabrics for data center interconnection networks," *Opt. Express*, vol. 23, no. 2, pp. 1159–1175, 2015. DOI: 10.1364/OE.23.001159. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-23-2-1159.

[44] L. Liu, R. Muñoz, R. Casellas, T. Tsuritani, R. Martínez, and I. Morita, "Openslice: An openflow-based control plane for spectrum sliced elastic optical path net-

works," in *2012 38th European Conference and Exhibition on Optical Communications*, 2012, pp. 1–3. DOI: 10.1364/ECEOC.2012.Mo.2.D.3.

[45] R. Casellas, R. Martinez, R. Muñoz, R. Vilalta, L. Liu, T. Tsuritani, and I. Morita, "Control and management of flexi-grid optical networks with an integrated stateful path computation element and openflow controller [invited]," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 5, A57–A65, Oct. 2013. DOI: 10.1364/JOCN.5.000A57.

[46] Z. Zhu, C. Chen, X. Chen, S. Ma, L. Liu, X. Feng, and S. J. B. Yoo, "Demonstration of cooperative resource allocation in an openflow-controlled multidomain and multinational sd-eon testbed," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1508–1514, 2015, ISSN: 1558-2213. DOI: 10.1109/JLT.2015.2389526.

[47] J. Yin, J. Guo, B. Kong, H. Yin, and Z. Zhu, "Experimental demonstration of building and operating qos-aware survivable vsd-eons with transparent resiliency," *Opt. Express*, vol. 25, no. 13, pp. 15 468–15 480, 2017. DOI: 10.1364/OE.25.015468. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-25-13-15468.

[48] Y. Shen, P. Samadi, Z. Zhu, A. Gazman, E. Anderson, D. Calhoun, M. Hattink, and K. Bergman, "Software-defined networking control plane for seamless integration of silicon photonics in datacom networks," in *2017 European Conference on Optical Communication (ECOC)*, 2017, pp. 1–3. DOI: 10.1109/ECOC.2017.8346132.

[49] Y. Shen, M. H. N. Hattink, P. Samadi, Q. Cheng, Z. Hu, A. Gazman, and K. Bergman, "Software-defined networking control plane for seamless integration of multiple silicon photonic switches in datacom networks," *Opt. Express*, vol. 26, no. 8, pp. 10 914–10 929, 2018. DOI: 10.1364/OE.26.010914. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-26-8-10914.

[50] Y. Shen, A. Gazman, Z. Zhu, M. Y. The, M. Hattink, S. Rumley, P. Samadi, and K. Bergman, "Autonomous dynamic bandwidth steering with silicon photonic-based wavelength and spatial switching for datacom networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, 2018, pp. 1–3.

[51] *What is openflow? definition and how it relates to sdn*, https://www.sdxcentral.com/networking/sdn/definitions/what-is-openflow/, (Accessed on 10/30/2019).

[52] *Wireshark · go deep.* https://www.wireshark.org/, (Accessed on 10/31/2019).

[53] *Sdn / openflow / message layer / flowmod | flowgrammable*, http://flowgrammable.org/sdn/openflow/message-layer/flowmod/, (Accessed on 10/31/2019).

[54] Y. Huang, Q. Cheng, N. C. Abrams, J. Zhou, S. Rumley, and K. Bergman, "Automated calibration and characterization for scalable integrated optical switch fabrics without built-in power monitors," in *2017 European Conference on Optical Communication (ECOC)*, 2017, pp. 1–3. DOI: 10.1109/ECOC.2017.8345829.

[55] Q. Cheng, M. Bahadori, Y. Huang, S. Rumley, and K. Bergman, "Smart routing tables for integrated photonic switch fabrics," in *2017 European Conference on Optical Communication (ECOC)*, 2017, pp. 1–3. DOI: 10.1109/ECOC.2017.8345828.

[56] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*, IEEE, 2008, pp. 77–88.

[57] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: Topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ACM, 2009, p. 41.

[58] B. Arimilli, R. Arimilli, N. Ni, R. Rajamony, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, *et al.*, "The percs high-performance interconnect," in *2010 18th IEEE Symposium on High Performance Interconnects*, IEEE, 2010, pp. 75–82.

[59] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Press, 2014, pp. 348–359.

[60] S. Rumley, D. Nikolova, R. Hendry, Q. Li, D. Calhoun, and K. Bergman, "Silicon photonics for exascale systems," *Journal of Lightwave Technology*, vol. 33, no. 3, pp. 547–562, 2015.

[61] Y. Xia, M. Schlansker, T. E. Ng, and J. Tourrilhes, "Enabling topological flexibility for data centers using omniswitch.," in *HotCloud*, 2015.

[62] C. Minkenberg, G. Rodriguez, B. Prisacari, L. Schares, P. Heidelberger, D. Chen, and C. Stunkel, "Performance benefits of optical circuit switches for large-scale dragonfly networks," in *Optical Fiber Communication Conference*, Optical Society of America, 2016, W3J–3.

[63] *Gtc*, http://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/gtc/, (Accessed on 11/3/2019).

[64] C. Camarero, C. Martinez, and R. Beivide, *IEEE Transactions on Parallel and Distributed Systems*,

[65] ——, "Random folded clos topologies for datacenter networks," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 193–204. DOI: 10.1109/HPCA.2017.26.

[66] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Holzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of Clos topologies and centralized control in Googleś datacenter network," in *Sigcomm '15*, 2015.

[67] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: Topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09, 2009, 41:1–41:11, ISBN: 978-1-60558-744-8. DOI: 10.1145/1654059.1654101.

[68] W. Jiang, J. Qi, J. X. Yu, J. Huang, and R. Zhang, "Hyperx: A scalable hypergraph framework," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2018, ISSN: 1041-4347. DOI: 10.1109/TKDE.2018.2848257.

[69] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The blackwidow high-radix clos network," in *33rd International Symposium on Computer Architecture (ISCA'06)*, 2006, pp. 16–28. DOI: 10.1109/ISCA.2006.40.

[70] N. Jain, A. Bhatele, L. H. Howell, D. Böhme, I. Karlin, E. A. León, M. Mubarak, N. Wolfe, T. Gamblin, and M. L. Leininger, "Predicting the performance impact of different fat-tree configurations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17, Denver, Colorado, 2017, 50:1–50:13, ISBN: 978-1-4503-5114-0. DOI: 10.1145/3126908.3126967.

[71] T. Hoefler, T. Schneider, and A. Lumsdaine, "Multistage switches are not crossbars: Effects of static routing in high-performance networks," in *2008 IEEE International Conference on Cluster Computing*, 2008, pp. 116–125. DOI: 10.1109/CLUSTR.2008.4663762.

[72] E. A. León, I. Karlin, A. Bhatele, S. H. Langer, C. Chambreau, L. H. Howell, T. D'Hooge, and M. L. Leininger, "Characterizing parallel scientific applications on commodity clusters: An empirical study of a tapered fat-tree," in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 909–920.

[73] A. Chatzieleftheriou, S. Legtchenko, H. Williams, and A. I. T. Rowstron, "Larry: Practical network reconfigurability in the data center," in *NSDI*, 2018.

[74] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, Aug. 2009, ISSN: 0146-4833. DOI: 10.1145/1594977.1592576. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592576.

[75] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013. DOI: 10.1145/2408776.2408794. [Online]. Available: https://doi.org/10.1145/2408776.2408794.

[76] X. Yang and Z. Lan, "Cooperative batch scheduling for hpc systems," 2016.

[77] F. Redaelli, M. D. Santambrogio, and D. Sciuto, "Task scheduling with configuration prefetching and anti-fragmentation techniques on dynamically reconfigurable systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '08, Munich, Germany, 2008, pp. 519–522, ISBN: 978-3-9810801-3-1. DOI: 10.1145/1403375.1403500.

[78] Y. Georgiou and M. Hautreux, "Evaluating scalability and efficiency of the resource and job management system on large hpc clusters," in *Job Scheduling Strategies for Parallel Processing*, W. Cirne, N. Desai, E. Frachtenberg, and U. Schwiegelshohn, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 134–156, ISBN: 978-3-642-35867-8.

[79] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. Sexton, and R. Walkup, "Optimizing task layout on the blue gene/l supercomputer," *IBM Journal on Research and Development*, vol. 49, 2005.

[80] C. Wang, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive process-level live migration in hpc environments," in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, ser. SC '08, Austin, Texas: IEEE Press, 2008, 43:1–43:12, ISBN: 978-1-4244-2835-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1413370.1413414.

[81] G. Michelogiannakis, Y. Shen, M. Y. Teh, X. Meng, B. Aivazi, T. Groves, J. Shalf, M. Glick, M. Ghobadi, L. Dennison, and K. Bergman, "Bandwidth steering in hpc using silicon nanophotonics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19, Denver, Colorado: ACM, 2019, 41:1–41:25, ISBN: 978-1-4503-6229-0. DOI: 10.1145/3295500.3356145. [Online]. Available: http://doi.acm.org/10.1145/3295500.3356145.

[82] G. Michelogiannakis, B. Aivazi, Y. Shen, L. Dennison, J. Shalf, K. Bergman, and M. Glick, "Architectural opportunities and challenges from emerging photonics in

future systems," in *2018 Photonics in Switching and Computing (PSC)*, 2018, pp. 1–3. DOI: 10.1109/PS.2018.8751354.

[83] *Ryu sdn framework*, https://osrg.github.io/ryu/, (Accessed on 11/02/2019).

[84] *Mpich | high-performance portable mpi*, https://www.mpich.org/, (Accessed on 11/04/2019).

[85] K. Madduri, K. Z. Ibrahim, S. Williams, E. Im, S. Ethier, J. Shalf, and L. Oliker, "Gyrokinetic toroidal simulations on leading multi- and manycore hpc systems," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–12. DOI: 10.1145/2063384.2063415.

[86] K. Z. Ibrahim, K. Madduri, S. Williams, B. Wang, S. Ethier, and L. Oliker, "Analysis and optimization of gyrokinetic toroidal simulations on homogenous and heterogenous platforms," *The International Journal of High Performance Computing Applications*, vol. 27, no. 4, pp. 454–473, 2013. DOI: 10.1177/1094342013492446. eprint: https://doi.org/10.1177/1094342013492446. [Online]. Available: https://doi.org/10.1177/1094342013492446.

[87] W. Deng, S. S. Iyengar, and N. E. Brener, "A fast parallel thinning algorithm for the binary image skeletonization," *The International Journal of High Performance Computing Applications*, vol. 14, no. 1, pp. 65–81, 2000. DOI: 10.1177/109434200001400105. eprint: https://doi.org/10.1177/109434200001400105. [Online]. Available: https://doi.org/10.1177/109434200001400105.

[88] Y. Shen, M. S. Glick, and K. Bergman, "Silicon photonic-enabled bandwidth steering for resource-efficient high performance computing," in *Metro and Data Center Optical Networks and Short-Reach Links II*, A. K. Srivastava, M. Glick, and Y. Akasaka, Eds., International Society for Optics and Photonics, vol. 10946, SPIE, 2019, pp. 17 –25. DOI: 10.1117/12.2509060. [Online]. Available: https://doi.org/10.1117/12.2509060.

[89] Y. Shen, S. Rumley, K. Wen, Z. Zhu, A. Gazman, and K. Bergman, "Accelerating of high performance data centers using silicon photonic switch-enabled bandwidth steering," in *2018 European Conference on Optical Communication (ECOC)*, 2018, pp. 1–3. DOI: 10.1109/ECOC.2018.8535322.

[90] *Tcpdump/libpcap public repository*, https://www.tcpdump.org/, (Accessed on 11/05/2019).

[91] *Iperf - the tcp, udp and sctp network bandwidth measurement tool*, https://iperf.fr/, (Accessed on 11/05/2019).

[92] A. Rylyakov, J. Proesel, S. Rylov, B. Lee, J. Bulzacchelli, A. Ardey, B. Parker, M. Beakes, C. Baks, C. Schow, and M. Meghelli, "22.1 a 25gb/s burst-mode receiver for rapidly reconfigurable optical networks," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1–3. DOI: 10.1109/ISSCC.2015.7063095.

[93] P. Samadi, M. Fiorani, Y. Shen, L. Wosinska, and K. Bergman, "Self-adaptive, multi-rate optical network for geographically distributed metro data centers," in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, 2017, pp. 1–3.

[94] ——, "Flexible architecture and autonomous control plane for metro-scale geographically distributed data centers," *Journal of Lightwave Technology*, vol. 35, no. 6, pp. 1188–1196, 2017, ISSN: 1558-2213. DOI: 10.1109/JLT.2017.2652480.

[95] M. Fiorani, P. Samadi, Y. Shen, L. Wosinska, and K. Bergman, "Flexible architecture and control strategy for metro-scale networking of geographically distributed data centers," in *ECOC 2016; 42nd European Conference on Optical Communication*, 2016, pp. 1–3.

[96] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10, Melbourne, Australia: ACM, 2010, pp. 267–280, ISBN: 978-1-4503-0483-2. DOI: 10.1145/1879141.1879175. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879175.

[97] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," *Communications of the ACM*, vol. 54, pp. 95–104, Oct. 2009. DOI: 10.1145/1897852.1897877.

[98] T. Benson, A. Anand, A. Akella, and M. Zhang, "Microte: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11, Tokyo, Japan: ACM, 2011, 8:1–8:12, ISBN: 978-1-4503-1041-3. DOI: 10.1145/2079296.2079304. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079304.

[99] S. Yan, E. Hugues-Salas, V. J. F. Rancaňo, Y. Shu, G. M. Saridis, B. Rahimzadeh Rofoee, Y. Yan, A. Peters, S. Jain, T. May-Smith, P. Petropoulos, D. J. Richardson, G. Zervas, and D. Simeonidou, "Archon: A function programmable optical interconnect architecture for transparent intra and inter data center sdm/tdm/wdm networking," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1586–1595, 2015, ISSN: 1558-2213. DOI: 10.1109/JLT.2015.2392554.

[100] Gang Chen, Hongxiang Guo, D. Zhang, Yong Zhu, Cen Wang, Haomin Yu, Yusheng Wang, Jialiang Wang, Jian Wu, Xiaoyuan Cao, N. Yoshikane, T. Tsuritani, I. Morita, and M. Suzuki, "First demonstration of holistically-organized metro-

embedded cloud platform with all-optical interconnections for virtual datacenter provisioning," in *2015 Opto-Electronics and Communications Conference (OECC)*, 2015, pp. 1–3. DOI: 10.1109/OECC.2015.7340294.

[101] R. Doverspike, G. Clapp, P. Douyon, D. M. Freimuth, K. Gullapalli, B. Han, J. Hartley, A. Mahimkar, E. Mavrogiorgis, J. O'Connor, J. Pastor, K. K. Ramakrishnan, M. E. Rauch, M. Stadler, A. Von Lehmen, B. Wilson, and S. L. Woodward, "Using sdn technology to enable cost-effective bandwidth-on-demand for cloud services [invited]," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, no. 2, A326–A334, 2015, ISSN: 1943-0639. DOI: 10.1364/JOCN.7.00A326.

[102] D. Adami, B. Martini, A. Sgambelluri, M. Gharbaoui, P. Castoldi, C. Callegari, L. Donatini, and S. Giordano, "Cloud and network service orchestration in software defined data centers," in *2015 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, 2015, pp. 1–6. DOI: 10.1109/SPECTS.2015.7285284.

[103] Y. Shen, P. Samadi, and K. Bergman, "Autonomous network and it resource management for geographically distributed data centers," *J. Opt. Commun. Netw.*, vol. 10, no. 2, A225–A231, 2018. DOI: 10.1364/JOCN.10.00A225. [Online]. Available: http://jocn.osa.org/abstract.cfm?URI=jocn-10-2-A225.

[104] F. P. Tso, S. Jouet, and D. P. Pezaros, "Network and server resource management strategies for data centre infrastructures: A survey," *Computer Networks*, vol. 106, pp. 209 –225, 2016, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2016.07.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128616302298.

[105] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, "Vmflow: Leveraging vm mobility to reduce network power costs in data centers," in *NETWORKING 2011*, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 198–211, ISBN: 978-3-642-20757-0.

[106] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, "Remedy: Network-aware steady state vm management for data centers," in *NETWORKING 2012*, R. Bestak, L. Kencl, L. E. Li, J. Widmer, and H. Yin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 190–204, ISBN: 978-3-642-30045-5.

[107] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *2011 Proceedings IEEE INFOCOM*, 2011, pp. 71–75. DOI: 10.1109/INFCOM.2011.5935254.

[108] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Network*, vol. 29, no. 5, pp. 36–42, 2015, ISSN: 1558-156X. DOI: 10.1109/MNET.2015.7293303.

[109] M. Schiano, A. Percelsi, and M. Quagliotti, "Flexible node architectures for metro networks," in *2015 Optical Fiber Communications Conference and Exhibition (OFC)*, 2015, pp. 1–3. DOI: 10.1364/OFC.2015.W3J.4.

[110] Gang Chen, Hongxiang Guo, D. Zhang, Yong Zhu, Cen Wang, Haomin Yu, Yusheng Wang, Jialiang Wang, Jian Wu, Xiaoyuan Cao, N. Yoshikane, T. Tsuritani, I. Morita, and M. Suzuki, "First demonstration of holistically-organized metro-embedded cloud platform with all-optical interconnections for virtual datacenter provisioning," in *2015 Opto-Electronics and Communications Conference (OECC)*, 2015, pp. 1–3. DOI: 10.1109/OECC.2015.7340294.

[111] M. Fiorani, S. Aleksic, P. Monti, J. Chen, M. Casoni, and L. Wosinska, "Energy efficiency of an integrated intra-data-center and core network with edge caching," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 6, no. 4, pp. 421–432, 2014, ISSN: 1943-0639. DOI: 10.1364/JOCN.6.000421.

[112] *Inphi debuts 100g dwdm solution for 80km data center interconnects - inphi*, https://www.inphi.com/media-center/press-releases/inphi-debuts-100g-dwdm-solution-for-80km-data-center-interconnects/, (Accessed on 12/02/2019).

[113] E. Burmeister, D. Blumenthal, and J. Bowers, "A comparison of optical buffering technologies," *Optical Switching and Networking*, vol. 5, no. 1, pp. 10 –18, 2008, Special Section: Photonics in Switching 2006, ISSN: 1573-4277. DOI: https://doi.org/10.1016/j.osn.2007.07.001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1573427707000264.